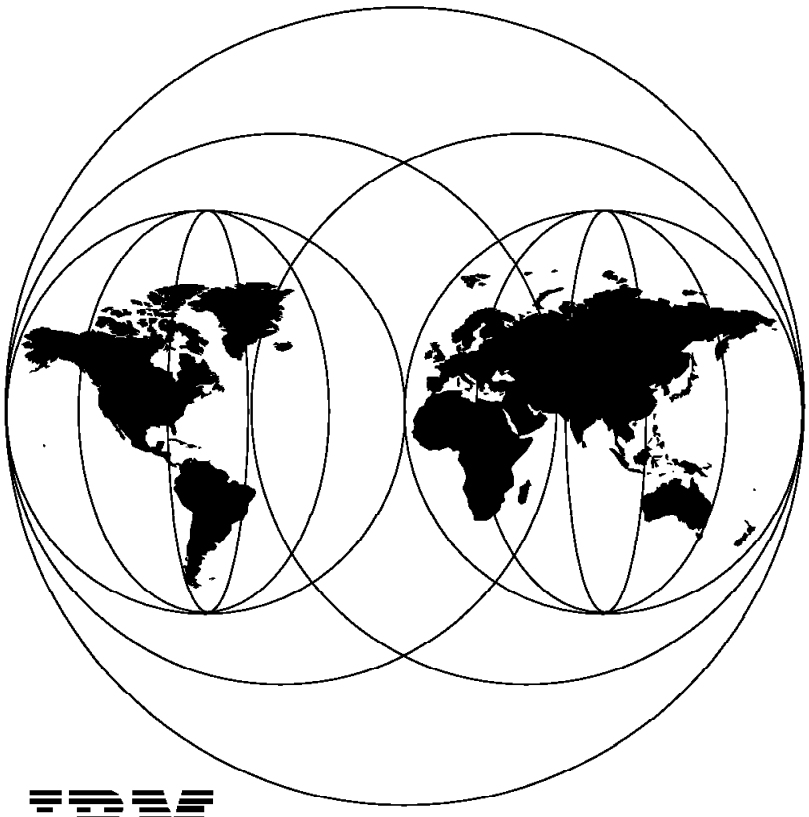
International Technical Support Organization

SG24-4536-00

CICS/ESA-DB2 Interface Guide

September 1995



IBM

International Technical Support Organization
San Jose Center

International Technical Support Organization

SG24-4536-00

CICS/ESA-DB2 Interface Guide

September 1995

Take Note!

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

First Edition (September 1995)

This book is based upon the CICS-IBM DATABASE 2 Interface Guide, GG24-3202-1. This book replaces makes obsolete GG24-3202-1.

This edition applies to Version 3, Release 3 of the IBM licensed program Customer Information Control System/Enterprise Systems Architecture (CICS/ESA), program number 5685-083, to Version 4, Release 1 of the IBM licensed program Customer Information Control System/Enterprise Systems Architecture (CICS/ESA), program number 5655-018, to Version 2, Release 3 of IBM DATABASE 2 (DB2), program number 5665-DB2, and to Version 3, Release 1 of IBM DATABASE 2 (DB2), program number 5685-DB2.

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

An ITSO Technical Bulletin Evaluation Form for reader′s feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 471 Building 070B
5600 Cottle Road
San Jose, California 95193-0001

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

Abstract

This document provides details about the interface between CICS/ESA and DB2. It focuses on considerations for running CICS application programs that use the CICS attachment facility and on the planning, installation, and tuning of the interface between CICS/ESA and DB2. The document also provides examples of implementations of the interface between CICS/ESA and DB2.

This document was written for customers and technical professionals who need to know how to take advantage of the CICS attachment facility. Some knowledge of CICS/ESA, DB2, and MVS/ESA is assumed.

(239 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help customers and technical professionals to understand, evaluate, and implement the CICS-DB2 interface. The information in this publication is not intended as the specification of any programming interfaces that are provided by CICS/ESA or DB2. See the PUBLICATIONS section of the IBM Programming Announcements for CICS/ESA and DB2 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to program temporary fix (PTF) numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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

CICS	CICS/ESA
CICS/MVS	DATABASE 2
DB2	IBM
IMS/ESA	MVS/ESA
NetView	RACF

VTAM

The following terms are trademarks of other companies:

Windows is a trademark of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

C-bus is a trademark of Corollary, Inc.

Other trademarks are trademarks of their respective companies.

Preface

We organized this book so that you can select one or more topics without having to read through the whole document. We assume that you are familiar with the functions of CICS and DB2.

The contents are based on:

- IBM DATABASE 2 Version 2 Release 3 (5740-XYR), called DB2
- IBM DATABASE 2 Version 3 Release 1 (5685-DB2), called DB2
- CICS/ESA Version 3 Release 3 (5685-083), called CICS
- CICS/ESA Version 4 Release 1 (5655-018), called CICS.

We use the terms CICS and DB2 when there is no need to distinguish between the different versions of these products. However, the DB2 interface differs between the two versions of CICS. We use the term CICS/ESA 3.3 or CICS/ESA 4.1 to distinguish between the two versions of CICS. Similarly we use DB2 2.3 or DB2 3.1 to distinguish between the two versions of DB2.

The resource manager interface (RMI) mentioned in the DB2 documentation is exactly the same function as the task related user exit (TRUE) interface. Both notations are used in this guide.

How This Document Is Organized

The chapters in this document provide the following information:

- Chapter 1, “Introduction to the CICS Attachment Facility”

  In this chapter we give an overview of the CICS attachment facility, which provides a multithread connection. Each CICS transaction accessing DB2 uses a different MVS task control block (TCB), thus exploiting a multiprocessor’s capability for overlapped processing. The architecture allows a CICS task to access both DB2 and CICS resources.

- Chapter 2, “Installation”

  In this chapter we describe the tasks you must complete to install the CICS-DB2 interface. You must define several CICS resources (transactions and programs) to install the interface. You must also customize your system initialization table (SIT). You may, optionally, modify your program list table (PLT) to include attachment entries. You must define a new table, the resource control table (RCT). The RCT defines the threads of the connections and how they are used by different CICS transactions. During installation you also define the connection security between CICS and DB2.

- Chapter 3, “Operation”

  In this chapter we descibe how to operate a CICS-DB2 system, including functions like establishing and stopping the connection. We also briefly introduce monitoring the CICS attachment facility, which we describe fully in Chapter 7, “Monitoring, Tuning, and Handling Deadlocks” on page 123.

- Chapter 4, “Security”

  In this chapter we describe how you can use the IBM program product, Resource Access Control Facility (RACF) to protect DB2 databases and data sets and to verify that CICS is authorized to connect to DB2. CICS provides additional security functions for checking the transaction authorization, and the user authentication. You can use DB2 security to restrict access to DB2 resources to authorized users only.

- Chapter 5, “Defining the Resource Control Table”

  In this chapter we describe how the RCT is used to define the connection between CICS and DB2. The definition is made up of a number of parameters. We provide hints and tips for defining the RCT, as well as a series of examples. We also describe how different thread types are created, used, and terminated.

- Chapter 6, “Program Preparation and Testing”

  In this chapter we describe:

  – The different environments where programs can be prepared and tested
  – The steps required for program preparation and testing
  – The bind process and the consequences of different bind options
  – The tools available for program testing and debugging
  – The considerations for moving an application from test to production.

- Chapter 7, “Monitoring, Tuning, and Handling Deadlocks”

  In this chapter we describe:

  – The monitoring tools provided by the CICS attachment facility, DB2, and CICS
  – How to monitor the CICS attachment facility and CICS applications that access DB2
  – How to tune a CICS application and the CICS attachment facility
  – How to deal with a deadlock situation.

- Chapter 8, “Accounting”

  In this chapter we describe how you can use data produced by CICS and DB2 for accounting.

- Chapter 9, “CICS-DB2 Recovery and Restart”

  In a CICS-DB2 environment, a transaction can update resources controlled by CICS and resources controlled by DB2. CICS and DB2 control and recover their own resources. Nevertheless, DB2 may need to coordinate its recovery with that done by CICS. This coordination is implemented using the two-phase commit protocol (TPCP), which we explain in this chapter.

- Chapter 10, “CICS-DB2 Application Design Considerations”

  This chapter contains information primarily for CICS application developers, CICS application reviewers, and people involved in defining standards for application design. In this chapter we discuss:

  – The importance of a consistent application design
  – CICS-DB2 design criteria
  – How to switch DB2 plans within an application
  – Programming considerations that are unique to the CICS-DB2 environment.

Recommended Reading

Figure 1 recommends which chapters you should read, depending on your role. An **R** indicates that you should read this chapter. An **O** indicates that reading is recommended, but optional.

Chapter	CICS Syst. Programmer	DB2 Syst. Programmer	DB2 DB Admin.	Systems Admin.	Appl. Designer	Operations
1	R	R	R	R	R	O
2	R	R		R		O
3	O	R	O	R	O	R
4	O	O	R	R	O	O
5	R	R	O	R	O	
6	O	O	R	R	R	
7	R	R	R	R		O
8	R	R	O	R	O	
9	R	R	R	O	R	O
10	O	O	R	R	R	

Figure 1. Recommended Reading Matrix

Related Publications

The publications listed in this section contain more detailed discussions of the topics covered in this document.

DB2 Version 3 Related Publications

- *IBM DATABASE 2 Administration Guide*, SC26-4888
- *IBM DATABASE 2 Application Programing and SQL Guide*, SC26-4889
- *IBM DATABASE 2 Command and Utility Reference*, SC26-4891
- *IBM DATABASE 2 Diagnosis Guide and Reference*, LY27-9603
- *IBM DATABASE 2 Messages and Codes*, SC26-4892
- *IBM DATABASE 2 SQL Reference*, SC26-4890

DB2 Version 2 Related Publications

- *IBM DATABASE 2 Administration Guide*, SC26-4374
- *IBM DATABASE 2 Application Programing and SQL Guide*, SC26-4377
- *IBM DATABASE 2 Command and Utility Reference*, SC26-4378
- *IBM DATABASE 2 Diagnosis Guide and Reference*, LY27-9536
- *IBM DATABASE 2 Messages and Codes*, SC26-4379
- *IBM DATABASE 2 SQL Reference*, SC26-4380

CICS/ESA Version 3 Related Publications

- *CICS/ESA Application Programming Guide*, SC33-0675
- *CICS/ESA Application Programming Reference*, SC33-0676
- *CICS/ESA CICS-RACF Security Guide*, SC33-0749
- *CICS/ESA Customization Guide*, SC33-0665
- *CICS/ESA Installation Guide*, SC33-0663
- *CICS/ESA Operations Guide*, SC33-0668
- *CICS/ESA Performance Guide*, SC33-0659
- *CICS/ESA Recovery and Restart Guide*, SC33-0658
- *CICS/ESA Resource Definition (Macro)*, SC33-0667
- *CICS/ESA Resource Definition (Online)*, SC33-0666
- *CICS/ESA System Definition Guide*, SC33-0664
- *CICS/ESA XRF Guide*, SC33-0661

CICS/ESA Version 4 Related Publications

- *CICS/ESA Application Programming Guide*, SC33-1169
- *CICS/ESA Application Programming Reference*, SC33-1170
- *CICS/ESA CICS-RACF Security Guide*, SC33-1185
- *CICS/ESA Customization Guide*, SC33-1165
- *CICS/ESA Installation Guide*, SC33-1163
- *CICS/ESA Messages and Codes*, SC33-1177
- *CICS/ESA Migration Guide*, GC33-1162
- *CICS/ESA Operations and Utilities Guide*, SC33-1167
- *CICS/ESA Performance Guide*, SC33-1183
- *CICS/ESA Recovery and Restart Guide*, SC33-1182
- *CICS/ESA Resource Definition Guide*, SC33-1166
- *CICS/ESA System Definition Guide*, SC33-1164

International Technical Support Organization Publications

Three of the redbooks published by the International Technical Support Organization are relevant to the CICS-DB2 interface:

- *DB2 Packages: Implementation and Use*, GG24-4001
- *DB2 Version 2.1 Security and Authorization Extensions Guide*, GG24-3299
- *CICS/MVS Overseer Program Customization Guide*, GG24-3319

A complete list of International Technical Support Organization publications, known as redbooks, with a brief description of each, may be found in:

> *International Technical Support Organization Bibliography of Redbooks,* GG24-3070.

To get a catalog of ITSO redbooks, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

A listing of all redbooks, sorted by category, may also be found on MKTTOOLS as ITSOPUB LISTALLX. This package is updated monthly.

How to Order ITSO Technical Publications

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their local IBM office. Guidance may be obtained by sending a PROFS note to BOOKSHOP at DKIBMVM1 or E-mail to bookshop@dk.ibm.com.

Customers may order hardcopy ITSO books individually or in customized sets, called GBOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain books on a variety of products.

ITSO Redbooks on the World Wide Web (WWW)

Internet users can find information about redbooks on the ITSO World Wide Web home page. To access the ITSO Web pages, point your Web browser (such as WebExplorer from the OS/2 3.0 Warp BonusPak) to the following:

```
http://www.redbooks.ibm.com/redbooks
```

IBM employees can access LIST3820 copies of redbooks as well. Point your Web browser to the IBM Redbooks home page:

```
http://w3.itsc.pok.ibm.com/redbooks/redbooks.html
```

Acknowledgments

This project was designed and managed by:

Hugh Smith
International Technical Support Organization, San Jose Center

The author of this document is:

Hector Pecina
IBM Argentina

The authors of the GG24-3202 edition of this document were:

Jan Egdoe
IBM Denmark

Claude Isoir
IBM France

The advisor for the GG24-3202 edition of this document was:

Lennart Aberg
IBM Sweden

This publication is the result of a residency conducted at the International Technical Support Organization, San Jose Center.

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Cathy Drummond
IBM Santa Teresa Laboratory

Dave Raiman
IBM Santa Teresa Laboratory

John Tilling
IBM United Kingdom Laboratories, Hursley UK

Thanks also to those who took the time to review this document, especially:

Marc Luong
IBM France

Chapter 1. Introduction to the CICS Attachment Facility

IBM DATABASE 2 (DB2) provides a relational database management system for the MVS environment. Attachment facilities provided by DB2 allow other MVS subsystems and batch address spaces to access DB2 resources through connections established using the MVS Subsystem Interface (SSI) protocol.

The attachment facilities provided by DB2 include the:

- ***CICS attachment facility***, which connects CICS to DB2.

  The CICS connection is called a single address space subsystem (SASS) connection. This connection allows multiple CICS users to access DB2, DL/I, and VSAM data concurrently.

- ***IMS/ESA attachment facility***, which connects an IMS/ESA subsystem, including an IMS/DB batch address space (a batch message processing program (BMP)) to access both DB2 and DL/I data.

  The IMS/ESA connection is called a multiple address space subsystem (MASS) connection because it allows connections from the IMS/ESA control region address space and from each dependent region address space.

- ***DB2-DL/I batch support***, which connects an MVS address space.

  The batch connection allows a batch program to access DB2, DL/I, or both DB2 and DL/I at the same time, with full recovery functions.

- ***TSO/batch attachment facility***, which connects a TSO subsystem or batch TSO address spaces to DB2.

- ***Call attachment facility*** (CAF), which allows an application program to access DB2.

  The CAF provides an alternate connection for TSO and batch applications needing tight control over the session environment. It is a low-level interface. For example, CAF does not provide full recovery functions. This is the responsibility of the application program. Use of this interface requires special application programming considerations.

Before a subsystem or a batch address space can access DB2 resources, it must:

1. Establish a connection to DB2.
2. Create one or more threads on the connection.

The connection establishes a communication path between the subsystem or batch address space and the DB2 subsystem.

Within the connection, a thread establishes a bidirectional path between a *user* in a subsystem or batch address space and specific *DB2 resources* (application plan or command processor).

1.1 Connection Types

DB2 supports two types of connections:

Single thread connections Only one thread is allowed per connection. This is the case for TSO and batch address spaces.

Multithread connections Multiple threads can be established between a connected subsystem and DB2. Each thread services the requests of a specific subsystem transaction. CICS and IMS/ESA use this connection type.

In CICS, there is a thread for each active CICS transaction accessing DB2.

In IMS/ESA, there is a thread associated with the control region and with each dependent region that accesses DB2.

Figure 2 shows the different connection types supported by DB2.

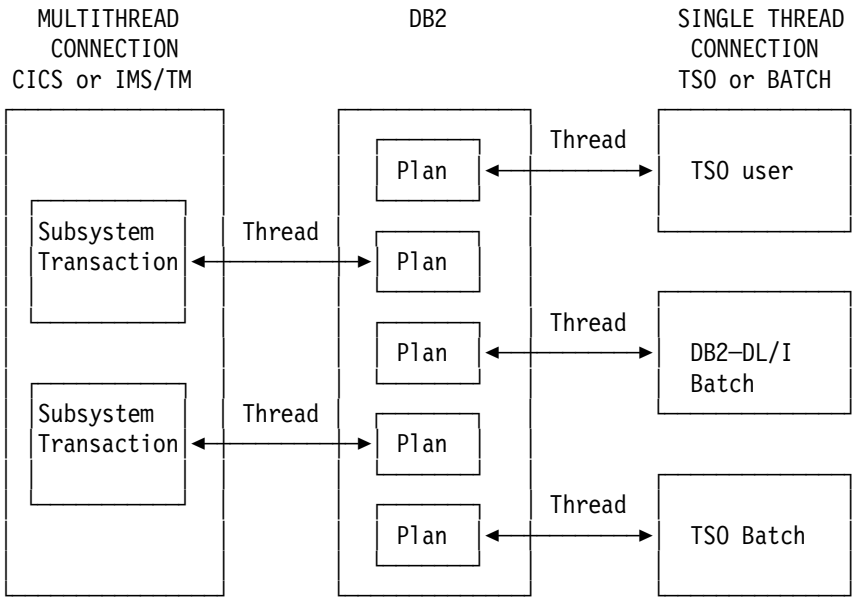


Figure 2. Subsystem and Batch Connections to DB2

1.2 The CICS Attachment Facility

The CICS attachment facility provides three major functions, an application program interface (API), attachment commands, and DB2 commands.

- ***Application program interface***. DB2 provides a language interface module that allows a CICS application program written in Assembler, C, COBOL, or PL/I to access DB2 databases using the data manipulating language (DML) subset of the Structured Query Language (SQL), or to define DB2 objects and control authority (GRANT and REVOKE) using the data description language (DDL) subset of SQL.

Access to DL/I and other CICS data can be intermixed with access to DB2 data. DB2 data is available only from programs using the CICS command-level interface, and EXEC SQL statements. If application programs access DB2 data, the application programs must be link-edited with the DB2 language interface module.

Updates to DB2 resources are fully synchronized with updates to CICS-protected resources such as file control, temporary storage, intrapartition transient data, and DL/I databases.

The CICS attachment facility controls the routing of SQL statements to DB2 and the synchronization of commit processing between the two subsystems through programs that use the CICS task related user exit (TRUE) function.

Note: SQL statements cannot be function shipped between CICS systems. You can access data from a DB2 system connected to a remote CICS system by routing the transaction to the remote CICS system, by using CICS distributed program link (DPL), or by using distributed transaction processing (DTP).

Two or more CICS systems can share the same DB2 system. On the other hand, each CICS system can be connected to only one DB2 subsystem at any one time.

Figure 3 on page 4 shows several CICS systems can be connected to DB2. Note, however, that all CICS systems accessing DB2 must run under the same MVS system as DB2. Even with DB2 Version 4, which allows multiple DB2 systems on different MVS images to access the same databases at the same time, a CICS region running on a particular MVS image can be connected only to a DB2 system running on the same MVS image.

There are several reasons why you may want to have separate CICS regions accessing the same DB2 system. For example:

- Your applications need different periods of operation.
- Extensions to existing systems, which already run in different CICS regions, can need to access the same DB2 data.
- Application growth causes you to split the application across more than one CICS region.

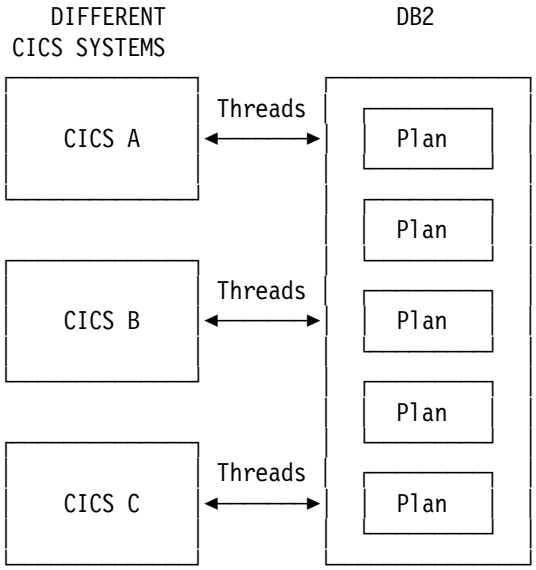


Figure 3. Three Different CICS Systems Connected to the Same DB2

- ***Attachment commands***. Attachment commands display and control the status of the attachment facility and are issued using the supplied CICS transaction DSNC.

  The attachment commands can be used to:

  - Start the connection to DB2 (STRT)
  - Stop the connection to DB2 (STOP)
  - Display the status of the connection to DB2 (DISP)
  - Modify characteristics of the connection to DB2 (MODI).

  You control the use of these CICS attachment facility commands through the standard CICS security mechanisms. The commands are not routed to DB2; so there is no authorization checking by DB2.

- ***DB2 commands***. Once a connection between CICS and DB2 is established, you can use DSNC to issue DB2 commands to the DB2 system, if you are authorized to use DSNC and to issue commands to DB2. The DB2 commands are routed to DB2 for processing. DB2 checks that the user has DB2 authority to issue the command. Responses are sent back to the originating user. These commands are used to display and control the status of the DB2 system.

  All DB2 commands entered through CICS must start with a dash (**-**), to show that the command is a DB2 command rather than an attachment command. Figure 46 on page 58 shows examples of attachment and DB2 commands.

1.2.1 Overview of the CICS-DB2 Attachment Facility

Figure 4 on page 5 shows an overview of the CICS attachment facility.

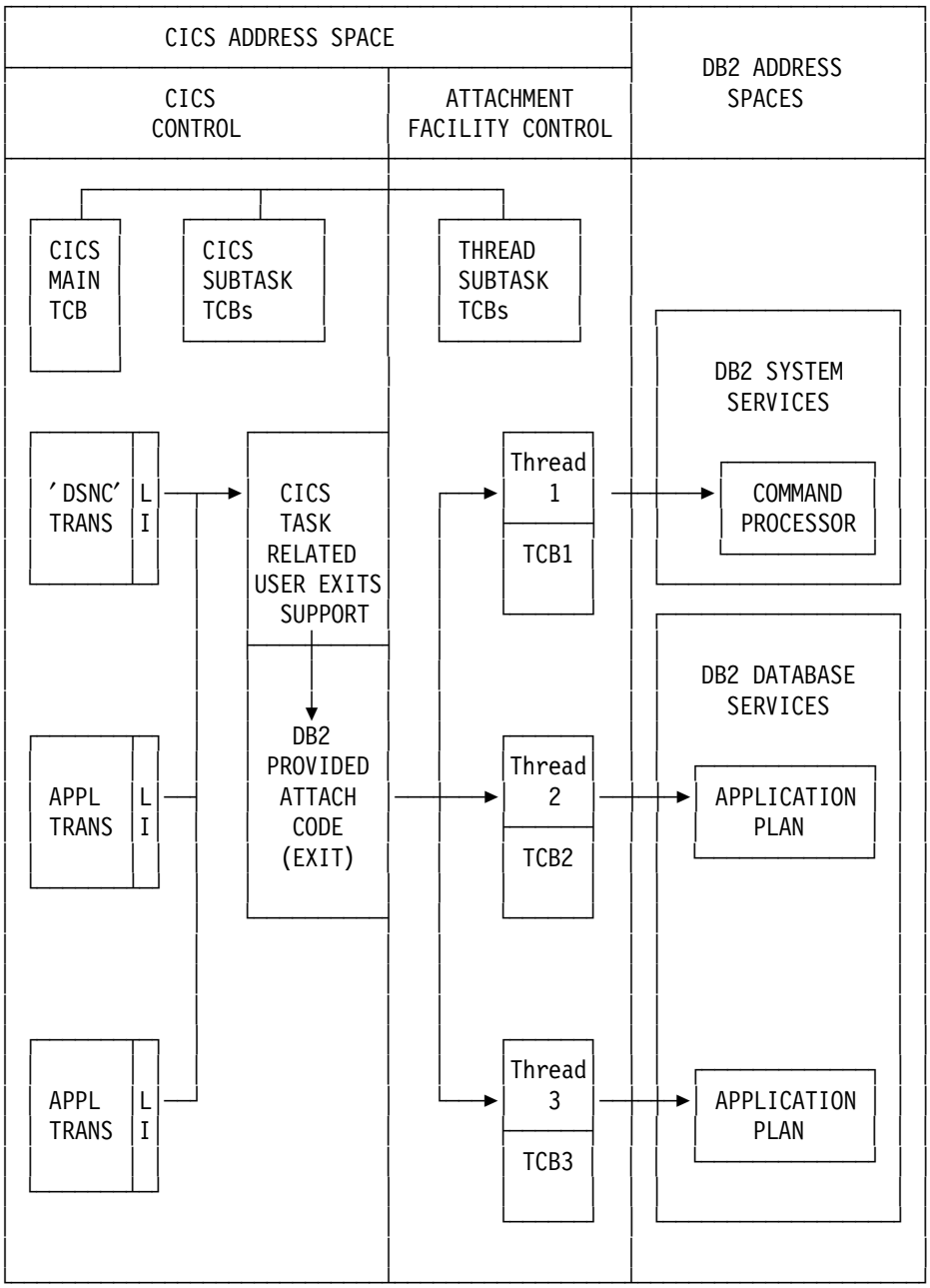


Figure 4. Overview of the CICS Attachment Facility

The CICS attachment facility provides a multithread connection to DB2. Each CICS application transaction and DB2 command uses a thread to access DB2 resources. These resources can be a command processor or an application plan.

CICS runs in its address space under a main task control block (TCB). Besides the main task, a number of subtasks can be run in the CICS address space. Each of these subtasks is executed under a subtask TCB. You can assign different priorities to the main task and subtasks, and they can be run in parallel, provided that your central electronic complex (CEC) has more than one processor. Each thread used by a CICS transaction, or by a command, runs under its own subtask TCB.

The CICS attachment facility provides three thread types: command threads, entry threads, and pool threads.

Command threads. The CICS attachment facility reserves some threads exclusively for attachment commands or DB2 commands. These are command threads. You specify how many threads are reserved for command usage. The CICS attachment facility reserves at least one command thread. If all the command threads are in use, and you want to issue another attachment or DB2 command (for example, during periods of heavy command usage), commands can overflow to the pool, using a pool thread.

Entry threads. You can reserve entry threads for use by specific DB2 plans and their associated CICS transactions. Entry threads are grouped together, and each group of entry threads serves one plan.[1] Each group of entry threads:

- Services one or more predefined transactions.
- Is associated with a *single* plan.[1] All transactions within a group of entry threads must use the same application plan.
- Consists of a user-specified number of entries. Each entry becomes a thread, if needed.

Based on user specifications, transactions from a group of entry threads can overflow to the pool (that is, use a pool thread) when the number of active transactions for the group of entry threads is greater than the maximum number of entry threads defined for the group.

An entry thread can be defined as *protected*. A protected entry thread is not terminated immediately if unused, but waits for another transaction from the same group of entry threads to use it. A protected entry thread is terminated if the CICS attachment facility finds that it is not in use at the end of a purge cycle, and the entry thread is not used during the next purge cycle (a purge cycle is 30-seconds, by default). This offers performance advantages.

Pool threads. You define a pool of threads for the connection. Pool threads are used by:

Command thread overflow

Entry thread overflow

Pool-defined transactions

The CICS attachment facility creates a pool thread when needed, and terminates the pool thread when no transaction is using it.

Refer to section 5.4.2, “TYPE=COMD Statement” on page 83 and section 5.4.3, “TYPE=POOL and TYPE=ENTRY Statement” on page 84 for detailed descriptions on how to define the threads, and to section 5.6, “Creating, Using, and Terminating Threads” on page 93 for descriptions of how the CICS attachment facility uses the threads during normal processing.

[1] You can associate a group of entry threads with multiple plans if you use dynamic plan selection. In this case you specify a program name as part of the definition of the entry thread. The program selects the plan that is to be used.

Figure 4 on page 5 shows three CICS transactions and the threads allocated to them. The first CICS transaction uses the command processing transaction (DSNC) provided by the CICS attachment facility. The other two transactions use application programs to access DB2 resources.

1.2.2 Program Flow through the CICS Attachment Facility

When an application program issues its first SQL request, the following events occur:

1. The DB2 language interface (LI in Figure 4 on page 5) invokes the CICS resource manager interface (RMI) program.
2. This formats the request and passes it to the attachment facility's task related user exit (TRUE) program.
3. The CICS attachment facility schedules a transaction thread.

   If a usable thread already exists it is used (unless it is already in use). If no threads exist, or all the existing threads are in use, a new thread is created. Each thread must have an MVS subtask TCB attached in the CICS address space. If necessary, the CICS attachment facility creates an MVS subtask TCB.

   During thread creation, DB2 checks authorization, creates control blocks, and allocates the application plan.
4. The thread TCB is posted when processing the SQL request, and the CICS task goes into a CICS WAIT state. The thread TCB uses cross-memory services to execute DB2 code to satisfy the SQL request. When the SQL request is completed, DB2 passes data back to the CICS attachment facility, sets any return codes, and returns control to the CICS attachment facility. The CICS task regains control and the thread TCB becomes inactive, remaining inactive (MVS WAIT state) until the CICS transaction issues another SQL request.
5. The CICS attachment facility returns control to the CICS application program.

This process, with step 3 omitted, is repeated for subsequent SQL requests from the same program.

When the program issues a syncpoint or terminates, the CICS attachment facility is invoked to synchronize commit processing between CICS and DB2.

DB2 commands entered from a CICS terminal are routed to DB2 similarly to the routing of SQL statements.

1.2.3 CICS Resource Control Table

The CICS DB2 resource control table (RCT) defines the connection between CICS and DB2. You create the RCT using a macro provided with the CICS attachment facility.

The communication control table (CCT) is built from the RCT during initialization of the CICS attachment facility. The CCT is an *internal* representation of the RCT. Each entry in the CCT is a possible thread. The two tables are illustrated in Figure 5 on page 8. The letters x, y, z, and n correspond to the values specified for the THRDM parameters in the RCT definitions.

Refer to 5.4, “Resource Control Table Parameters” on page 82 for a detailed description of how to define RCT parameters.

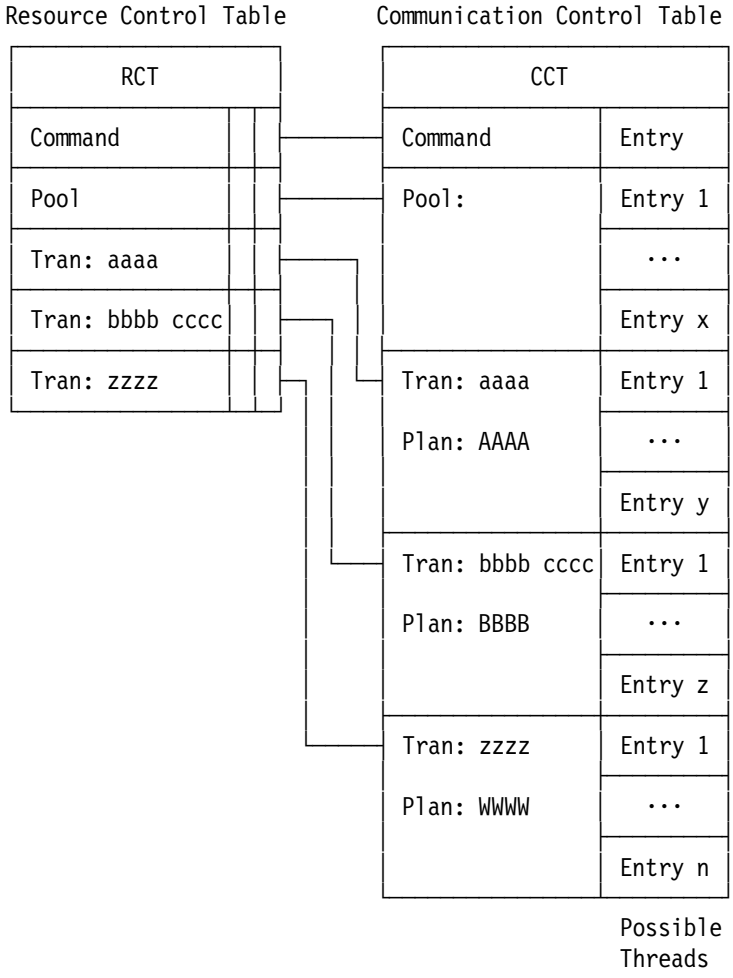


Figure 5. CICS Resource Control Table and Communication Control Table Relationship

1.3 The CICS-DB2 Communication Protocol

Figure 6 on page 9 represents the communication protocol for protected thread reuse between CICS and DB2 and shows the interaction between the two subsystems for the following sequence of events:

1. A connection between the CICS and DB2 subsystems is established.
2. User 1 starts a CICS task for a transaction from Group A (a group of entry threads). The transaction performs SQL statements and commits. CICS recoverable resources may also have been updated by this transaction.

   Unit of Recovery 1 (UR 1) represents the recoverable work performed by this transaction in DB2.
3. User 2 starts another CICS task, also for a transaction from Group A. The transaction performs SQL statements and commits. CICS recoverable resources may also have been updated by this transaction. UR 2 represents the unit of recovery for this transaction.

Note: The same thread is used to service two transactions, each representing a separate unit of recovery (UR).

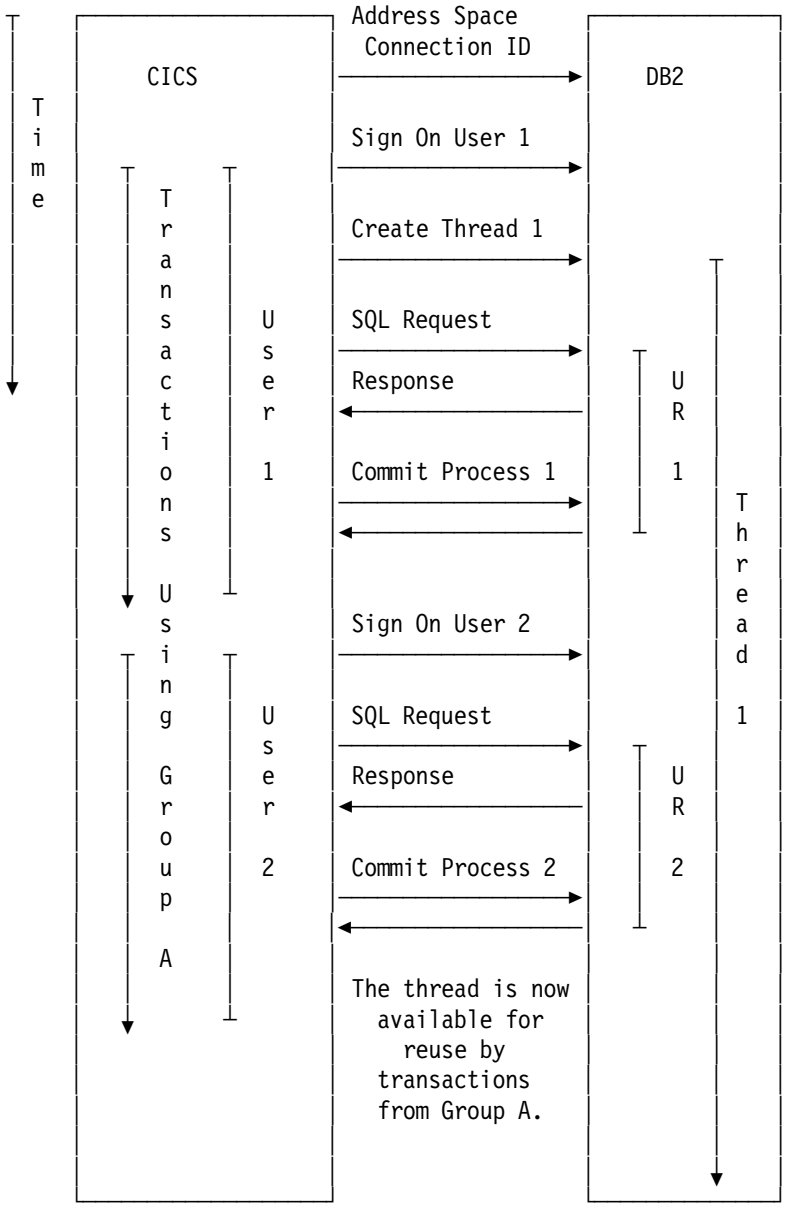


Figure 6. Communication Protocol for Protected Thread Reuse. If we were **not** *using a protected entry thread the thread would have been terminated at each task termination, and a new thread created for each task execution unless the second transaction was waiting for a thread when the first transaction released it.*

1.3.1 Address Space Connection, Verification, and Identification

An authorized CICS terminal user can start the connection by issuing a DSNC STRT command. Sections 3.2.1, “Starting the CICS Attachment Facility Manually (before CICS/ESA 4.1)” on page 48 and 3.2.2, “Starting the CICS Attachment Facility Manually (CICS/ESA 4.1 Onward)” on page 48 give details on starting and stopping the CICS attachment facility. The CICS attachment facility can also be started automatically as part of CICS initialization.

To verify that the CICS address space is authorized to access the DB2 subsystem, DB2 uses a security manager that is external to both CICS and DB2. The IBM external security manager is the Resource Access Control Facility (RACF). We refer to RACF specifically throughout this book, rather than a general external security manager. DB2 passes RACF the USERID on the JOB statement of the CICS startup JCL if CICS is started by submitting a job. If CICS is started with an MVS start command, the identifier is the ICHRIN03 entry in the RACF table for started procedures. Refer to section 4.1, “Resource Access Control Facility Connection Authorization” on page 61 for a more detailed description of connection authorization.

The *connection name* is an external name passed to DB2 to identify the connection when it is established. The connection name is the authorization ID from the RCT, or if the authorization ID is not specified in the RCT, the connection name is the CICS APPLID option specified in the system initialization table (SIT).

The CICS attachment facility does not allow CICS to connect to more than one DB2 subsystem at a time.

1.3.2 Thread Creation

When a transaction issues its first SQL statement, a DB2 thread is created, based on specifications in the RCT. A new thread is created if an existing one cannot be used.

A *correlation identification* (correlation ID) is assigned to each task that identifies the task uniquely. The correlation ID is a concatenation of the thread number used by the transaction and the CICS transaction ID. See Figure 89 on page 186 for a description of the correlation identification format.

1.3.3 Thread Termination

Threads can be terminated when no longer in use, depending on whether they are unprotected or protected.

- Unprotected threads are usually terminated at CICS task termination time.
- Protected threads that have not been reused during two consecutive purge cycles are terminated. The purge cycle time is 30 seconds for attachment facility code shipped before CICS/ESA 4.1. With CICS/ESA 4.1 the default value is 30 seconds, but you can use the new PURGEC parameter to alter the purge cycle time.

Threads can be forced to terminate. Reasons for this are:

- A DISCONNECT command is issued.
- If the number of thread subtask TCBs has passed THRDMAX-2, the CICS attachment facility can detach TCBs and terminate the corresponding threads. THRDMAX specifies the maximum number of threads the CICS attachment facility should create.
- The DB2 connection is terminated.

For a detailed description of thread creation and termination, see section 5.6, “Creating, Using, and Terminating Threads” on page 93.

1.3.4 Transaction Sign-on

This section describes the sign-on and sign-off process of transactions accessing DB2, and should not be confused with a user signing on or off from a terminal.

Sign-on is used only in multithread connections, such as CICS, to identify the thread user to DB2. When a new thread is created, the user's authorization ID is checked as part of the thread creation process. When a different user reuses an existing thread, the authorization ID is checked at thread allocation time.

The sign-on call in CICS provides an authorization ID. DB2 uses this authorization ID during thread creation to check that the user is authorized to access the requested DB2 plan or command.

The CICS attachment facility obtains the authorization ID from the RCT. The authorization ID can be:

- The 8-character user ID from sign-on
- The application name of the CICS system, specified by the APPLID option in the SIT
- The operator ID associated with the signed-on user (3 characters padded to 8)
- The terminal ID (4 characters padded to 8)
- The transaction ID (4 characters padded to 8)
- A user-defined character string (up to 8 characters)
- A RACF group added as a secondary DB2 authorization ID.

1.3.5 SQL Requests and DB2 Commands

Once the thread is created, DB2 accepts any SQL request or command that the supplied authorization ID is authorized to use. These include requests from an application program (using an entry thread or a pool thread) for:

- SQL DML statements to access or modify DB2 databases
- SQL DDL statements to define DB2 objects (databases, tables, and so on)
- GRANT and REVOKE statements for authorization control
- Requests using the DB2 Instrumentation Facility Interface (IFI).

Requests also include DB2 commands entered from a terminal (using a command or pool thread) to control and monitor the DB2 subsystem.

1.3.6 Synchronization of Commit Processing

The CICS attachment facility uses a CICS TRUE to ensure synchronization of commit processing between the two subsystems.

A UR is defined in DB2 as the updates performed by DB2 within a CICS logical unit of work (LUW). When CICS reaches a synchronization point, all updates within the UR must be committed in DB2. CICS uses the two-phase commit protocol to synchronize committing changes both to recoverable resources under its control and to DB2 resources. Refer to section 9.2, "Two-Phase Commit Protocol" on page 172 for a detailed description of the two-phase commit protocol. In two special circumstances CICS uses a single-phase commit protocol:

- If no updates are made in DB2, DB2 responds to the first phase of the commit protocol with a message informing CICS that there is no need for the second phase. This function was introduced by DB2 2.3.
- If only DB2 recoverable resources were updated, CICS automatically uses a single-phase commit process. This function was introduced by DB2 3.1.

Commit ends a unit of recovery. Changed data is made available to other applications and users. Note, however, that while DB2 page locks are freed at commit time, tablespace locks are freed at commit or thread termination time, depending on BIND parameters.

1.3.7 Connection Termination

Normal termination of the connection between CICS and DB2 occurs after one of the following events:

- An authorized CICS user issues a DSNC STOP (QUIESCE) command.
- CICS is terminated.
- DB2 is terminated.

During normal termination, each CICS transaction is allowed to complete its current UR before the thread subtask is detached.

Abnormal termination of the connection occurs due to a DSNC STOP FORCE command being issued, or abnormal termination of either CICS or DB2, and can result in incomplete URs. These are recovered automatically during the subsequent restart of the failed subsystem and the connection.

Table 1 gives a summary of the CICS-DB2 communication protocol described above.

Table 1 (Page 1 of 2). CICS-DB2 Communication Protocol Summary

Protocol Interaction	**CICS-DB2 Protocol**
Address space connection, verification, and identification	Connection is established by the operator entering a command, or at CICS startup. The CICS connection ID is specified in the RCT. If not, the APPLID from SIT is used as the connection name.
DB2 sign-on	Sign-on is done when creating a thread or reusing it with a new user ID. The authorization ID identifies the user of the thread. The authorization ID for CICS transactions is defined in the RCT and can be: • The 8-character user ID • The CICS application name • The CICS sign-on user ID • The CICS terminal name • The CICS transaction code • A user-defined character string • A Group added as secondary ID.

Table 1 (Page 2 of 2). CICS-DB2 Communication Protocol Summary	
Protocol Interaction	**CICS-DB2 Protocol**
Thread creation and termination	A thread is created following the first SQL call of a transaction, if there is not a thread that can be reused. Protected threads stay alive after usage, but are terminated when not used for an average of 45 seconds or for the PURGEC parameter. Unprotected threads are normally not reused, unless there is a transaction queued for the thread.
SQL requests and DB2 command requests	CICS transactions can issue SQL DDL/DML and GRANT/REVOKE statements, or make requests using the DB2 IFI. Command requests are issued from attachment code, and invoked through the DSNC transaction.
Commit protocol	CICS is the coordinator. The commit process is triggered by a CICS syncpoint. Backouts are also triggered by CICS (for example application abend). In a restart, CICS resolves DB2 in-doubt situations.
Connection termination	The connection is terminated by an operator command, CICS terminates, or DB2 terminates.

1.3.8 Extended Recovery Facility Considerations

With CICS/ESA you can use DB2 in an extended recovery facility (XRF) complex. In the DB2 documentation, DB2 is called *XRF tolerant*. This section gives an overview description of two different environments that use XRF, both with DB2 used as a database management system. One has a single central electronic complex (CEC), and one has two CECs. The full capabilities of CICS with XRF are described in the CICS documentation.

- ***CICS and DB2 in the same CEC***. with the XRF function, two CICS systems (the active and the alternate) are started under the same MVS. The active CICS system processing transactions is called the active, and the inactive one is called the alternate. If the active system fails, it disconnects from DB2. During the takeover process, the alternate CICS system performs an emergency restart. It then automatically connects to DB2, as described in section 3.2, “Starting the CICS Attachment Facility” on page 47, during the restart process. In-flight and in-abort transactions are backed out, to preserve data integrity. In-doubt transactions will be either backed out or committed, depending on information from CICS.
- ***Active CICS and alternate CICS in different CECs***. In a two-CEC environment with shared DASD, the DB2 databases can be accessed from only one complex at a time. To protect data from being corrupted, you must establish procedures to prevent DB2 from starting on more than one complex at a time. This restriction is removed by DB2 Version 4.

  With XRF, the active CICS and DB2 must run in the same CEC. If the active CICS goes down, it is disconnected from DB2 and DB2 *must* be stopped, if it does not fail too. You must ensure that when the alternate takes over from the active and becomes the new active, it does not connect to an existing DB2 subsystem started on the alternate CEC with the same subsystem ID as the DB2 subsystem previously on the active CEC. When the active and

alternate CICS are on different CECs, the DB2 system on the alternate CEC must be started after takeover and the connection then established. When the DB2 system on the new active CICS system is started, uncommitted updates in DB2 are backed out.

Refer to section 9.4.3.1, “Removed Recovery Restriction” on page 184 for more details about this process.

There are two different ways to stop or start DB2:

- Manually, by the operator
- Automatically, by a programmed operator.

In the latter case any automation product can be used, for example NetView, the Overseer program provided by CICS (refer to *CICS/MVS Overseer Program Customization Guide* for details), or any other product that suits your installation.

Chapter 2. Installation

In this chapter we describe the steps you must take to install the CICS attachment facility and to verify the installation using the sample transactions provided. We assume that both CICS and DB2 are already installed and operating.

2.1 Before You Start

Before you start implementing the CICS attachment facility you need to know which version of the CICS attachment facility you will use, who will implement it, and, if you are also using IMS, how you will implement the IMS Resource Lock Manager (IRLM).

2.1.1 Which Version of the CICS Attachment Facility?

IBM ships the CICS attachment facility modules and the macros used to define the characteristics of the connection as part of the DB2 installation tapes for DB2 2.3 and DB2 3.1. IBM ships a newer version of the CICS attachment facility with CICS/ESA 4.1. This newer CICS attachment facility takes advantage of features and enhancements in CICS/ESA 4.1. The CICS attachment facility shipped with CICS/ESA 4.1 works with all currently supported releases of DB2, but does not work with earlier releases of CICS. To connect to DB2 from CICS/ESA 4.1, you must use the new CICS attachment facility that is shipped with CICS/ESA 4.1. To connect to DB2 from CICS/ESA 3.3 or earlier releases of CICS, you must use the CICS attachment facility that is shipped with DB2.

The System Modification Program Extended (SMP/E) RECEIVE and APPLY jobs you used to install CICS and DB2 also installed all the CICS attachment facility modules and macros. In the following sections we describe the steps you must take to define and activate the connection and to verify the result of the installation using the sample programs provided. Figure 7 on page 16 shows the major tasks that need to be completed, and the person likely to complete them.

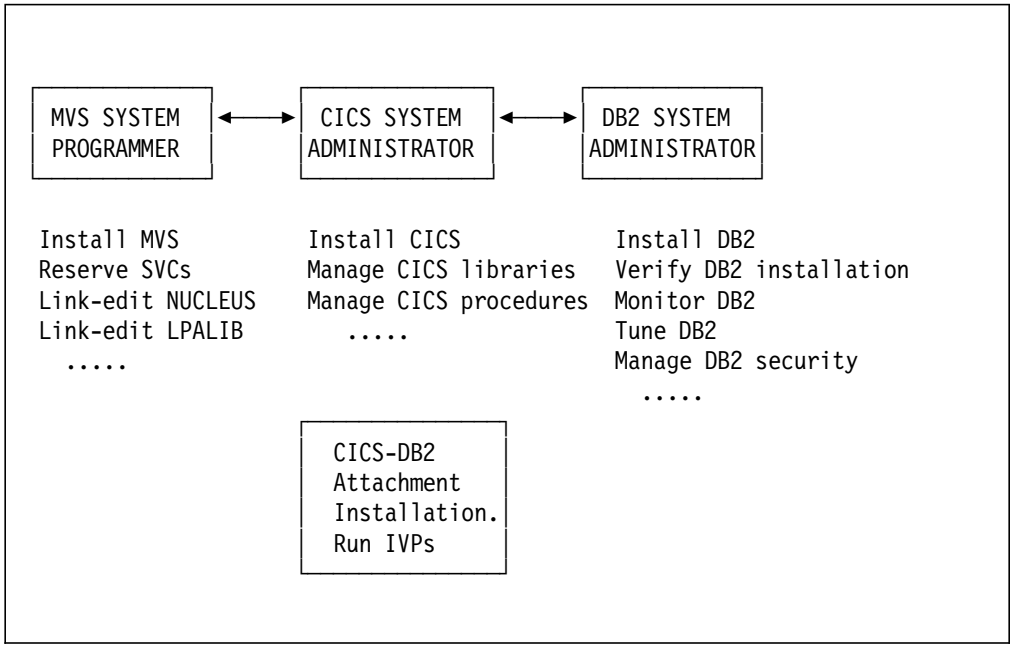


Figure 7. Personnel Involved in CICS Attachment Facility Installation

2.1.2 Who Installs the CICS Attachment Facility?

Our experience is that either the CICS system administrator or the CICS system programmer for your installation is normally responsible for installing the CICS attachment facility. This person must coordinate the installation activities with the DB2 system administrator and the MVS system programmer.

The MVS system programmer must give RACF authority to the CICS subsystem to connect to DB2.

The CICS system administrator must:

- Install the CICS attachment facility
- Run the installation verification procedures (IVPs).

The DB2 system administrator must provide the MVS system programmer with the following information and facilities to install the CICS attachment facility:

- The name of the DB2 subsystem specified in DB2 installation. The default name is DSN.
- The library data set name prefix and suffix specified when DB2 was installed (if you wish to use the CICS attachment facility modules shipped with DB2). The default prefix name for DB2 2.3 libraries is DSN230. The default prefix name for DB2 3.1 libraries is DSN310. Note that this prefix name is release sensitive.

  CICS execution procedures must refer to the CICS attachment facility modules in DSN310.SDSNLOAD or DSN230.DSNLOAD to access DB2 resources.
- The library data set name prefix and suffix specified when CICS/ESA 4.1 was installed (if you wish to use the CICS attachment facility modules shipped with CICS/ESA 4.1). The default prefix name for these libraries is CICS410.

CICS execution procedures must reference the CICS attachment facility modules in CICS410.SDFHLOAD to access DB2 resources.

- The procedures used to compile application programs
- Guidance on using the IVPs for CICS.
- Authorizations for the CICS attachment facility IVPs.

2.1.3 IMS Resource Lock Manager Considerations

The IMS Resource Lock Manager (IRLM) provides the locking function for DB2. You must use an IRLM when you use DB2. You may already be using an IRLM for an IMS system, for example, if you are using IMS Database Control (DBCTL), or if you are using IMS block level data sharing, or if you have chosen to use the IRLM for locking rather than the program isolation (PI) function of IMS. Each DB2 or IMS system can have its own IRLM, or two or more DB2 or IMS systems can share an IRLM.

We recommend that each subsystem (DB2 and IMS) has its own IRLM because:

- Using only one IRLM can lead to a large queue of resources to be handled for all subsystems.
- Having one IRLM for each DB2 subsystem allows you to stop a specific DB2 by cancelling its associated IRLM. This action should be carried out only under extreme circumstances, such as when the entire DB2 system has stalled.

2.2 Software Planning

If you wish to connect to DB2 from CICS/ESA 4.1, you must use the CICS attachment facility that is shipped with CICS/ESA 4.1. The older CICS attachment facility is still shipped with the DB2 product in order to continue support for CICS/MVS Version 2 and CICS/ESA Version 3.

Refer to the Program Directories for CICS and DB2 to verify prerequisites of related product levels and PTFs. You should also check with the IBM Support Center for any new installation or service information.

2.2.1 DB2 Libraries Used in the Installation

If you are using CICS/ESA Version 4, you need the following CICS and DB2 libraries to install the CICS attachment facility:

- CICS410.SDFHLOAD and CICS410.SDFHAUTH, which contain the CICS attachment facility modules
- CICS410.SDFHMAC, which contains the RCT macro
- CICS410.SDFHSAMP, which contains the dynamic plan user exit, DSNCUEXT
- DSN310.SDSNSAMP or DSN230.SDSNSAMP, which contains the DB2 sample application programs source for your release of DB2.

If you are using DB2 Version 3, you need the following DB2 libraries to install the CICS attachment facility:

- DSN310.SDSNLOAD, which contains the DB2 3.1 CICS attachment facility modules
- DSN310.SDSNMACS, which contains the DB2 3.1 RCT macro

- DSN310.SDSNSAMP, which contains the DB2 3.1 sample application programs source code, sample JCL for program preparation, sample CICS program definitions, sample CICS transaction definitions, a sample job for preparing the RCT, and the DSNTIJSU job used for implementing the CICS attachment facility.

If you are using DB2 Version 2, you need the following DB2 libraries to install the CICS attachment facility:

- DSN230.DSNLOAD, which contains the DB2 2.3 CICS attachment facility modules
- DSN230.DSNMACS, which contains the DB2 2.3 RCT macro
- DSN230.DSNSAMP, which contains the DB2 2.3 sample application programs source code, sample JCL for program preparation, sample CICS program definitions, sample CICS transaction definitions, a sample job for preparing the RCT, and the DSNTIJSU job used for implementing the CICS attachment facility.

You still need access to the DB2 libraries, even if you are implementing the CICS/ESA 4.1 version of the CICS attachment facility. Figure 8 on page 19 shows how these libraries would be included in the JCL to start a CICS/ESA 3.3 region.

```
//jobname  JOB ...,...
// INDEX1='CICS330',
// INDEX2='DSN310'
//CICS330 PROC
//CICS330J  EXEC PGM=DFHSIP,PARM=('DSNCRCT1,SYSIN'),REGION=5000K
//STEPLIB   DD DSN=&INDEX1.SDFHAUTH,DISP=SHR
//          DD DSN=&INDEX2.SDSNLOAD,DISP=SHR
//*         (ATTACHMENT LOAD MODULES)
//          DD DSN=&INDEX2.RCT.LOAD,DISP=SHR
//*         (RCT)
//DFHRPL    DD DSN=&INDEX1.SDFHLOAD,DISP=SHR
//          DD DSN=&INDEX1.PROGRAM,DISP=SHR
//*         (TABLES, APPLIC. PGMS, MAPS)
//          DD DSN=&INDEX2.SDSNLOAD,DISP=SHR
//*         (ACCESS TO DSNCCOM1, ETC...)
//          DD DSN=&INDEX2.RUNLIB.LOAD,DISP=SHR
//*         (ACCESS TO DB2-CICS SAMPLE PROGRAMS))
//          DD DSN=SYS1.COB2CICS,DISP=SHR
//          DD DSN=SYS1.COB2LIB,DISP=SHR
//          DD DSN=PLI.PLILINK,DISP=SHR
//*      (ACCESS TO 'DFHSAP' CICS MODULE WHEN PL/I USED)
//SYSUDUMP  DD SYSOUT=T
//ddname1   DD other dataset names needed
//       PEND
//CICS    EXEC CICS330
//CICS330J.SYSIN DD *
 SIT=DB
 APPLID=STCICSA
 START=AUTO
 ISC=YES
 IRCSTRT=YES
 PLTPI=TI
 PLTSD=TT
 .END
/*
//
```

Figure 8. Example of CICS JCL. This example shows JCL for a CICS/ESA 3.3 region with DB2 3.1.

2.3 CICS Attachment Facility Installation (CICS/ESA Version 3)

Perform the following required tasks to install the CICS attachment facility:

- Link-edit the CICS attachment facility load modules.
- Provide CICS access to the attachment facility modules.
- Define the CICS connection to DB2.
- Update CICS tables.
- Coordinate CICS and DB2 security.
- Verify the installation.

The following section includes a brief description of each of these tasks.

2.3.1 Link-editing CICS Attachment Facility Load Modules

DB2 provides the load modules for the CICS attachment facility. However, these modules have not been linkedited with the CICS stub modules. Before you can use the CICS attachment facility you must include the appropriate CICS load modules in your DB2 load library. The DB2 load library is either "DSN310.SDSNLOAD" (for DB2 Version 3) or "DSN230.DSNLOAD" (for DB2 Version 2). The IBM-supplied job, DSNTIJSU, performs the required link-edits for you. You can find DSNTIJSU in either DSN310.SDSNSAMP (DB2 Version 3) or DSN230.DSNSAMP (DB2 Version 2). If you are using CICS/ESA 3.2.1 (or a later release of CICS), DSNTIJSU also loads program and transaction definitions for the CICS attachment facility into your CICS system definition (CSD) file. You can customize DSNTIJSU (for example, by altering the STEPLIB statement to point to your CICS loadlib), but *do not change the ORDER statements*. DFHEAI must be the first module in the ORDER statement.

The modules that are link-edited are:

DSNCCOM0
DSNCCOM1
DSNCCOM2
DSNCEDF1
DSNCEDON
DSNCEXT1
DSNCEXT2
DSNCMSG0
DSNCSM31
DSNCSM21
DSNCSTOP
DSNCSTRT
DSNCUEXT

DSNCCOM0 is used only to start the CICS attachment facility. DSNCCOM0 is shipped with DB2 2.3 with APAR PN29087 applied, and later versions of DB2. DSNCCOM1 is used only for command processing with DB2 2.3 with APAR PN29087 applied, and later versions of DB2. Module DSNCCOM2 is used to stop the CICS attachment facility. See Chapter 3, "Operation" for information on how DSNCCOM1 and DSNCCOM2 are used.

Figure 9 on page 21 shows the statements from DSNTIJSU that link-edit the CICS attachment facility modules.

```
//SSRENT EXEC PGM=IEWL,
//        REGION=1024K,
// PARM='SIZE=(900K,124K),XREF,LET,RENT,NCAL,AMODE=24,RMODE=24'
//SYSPRINT DD SYSOUT=*
//DSNALOAD DD DISP=SHR,
//            DSN=DSN!!0.ADSNLOAD
//DSNLOAD  DD DISP=SHR,
//            DSN=DSN!!0.SDSNLOAD
//*  DLOADLIB IS USED FOR DFHEAI AND DFHEAIO CICS MODULES
//DLOADLIB DD DISP=SHR,
//            DSN=CICS170.LOADLIB
//SYSLMOD  DD DISP=SHR,
//            DSN=DSN!!0.SDSNLOAD
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLIN   DD  *
ORDER DFHEAI,DFHEAIO,DSNCCOMO,DSNAA
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAIO)
INCLUDE DSNLOAD(DSNCCOMO)
ENTRY DSNCCOMO
NAME DSNCCOMO(R)
ORDER DFHEAI,DFHEAIO,DSNCCOM1,DSNCCMDP,DSNAA
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAIO)
INCLUDE DSNLOAD(DSNCCOM1)
ENTRY DSNCCOM1
NAME DSNCCOM1(R)
ORDER DFHEAI,DFHEAIO,DSNCCOM2,DSNAA
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAIO)
INCLUDE DSNLOAD(DSNCCOM2)
ENTRY DSNCCOM2
NAME DSNCCOM2(R)
ORDER DFHEAI,DFHEAIO,DSNCEXT1,DSNAA
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAIO)
INCLUDE DSNLOAD(DSNCEXT1)
ENTRY DSNCEXT1
NAME DSNCEXT1(R)
ORDER DFHEAI,DFHEAIO,DSNCEXT2,DSNAA,DSNARIB
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAIO)
INCLUDE DSNLOAD(DSNCEXT2)
ENTRY DSNCEXT2
NAME DSNCEXT2(R)
ORDER DFHEAI,DFHEAIO,DSNCMSGO,DSNAA
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAIO)
INCLUDE DSNLOAD(DSNCMSGO)
ENTRY DSNCMSGO
NAME DSNCMSGO(R)
```

Figure 9 (Part 1 of 2). Link-editing Step from DB2 3.1 Version of DSNTIJSU

```
ORDER DFHEAI,DFHEAI0,DSNCSTOP,DSNAA
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAI0)
INCLUDE DSNLOAD(DSNCSTOP)
ENTRY DSNCSTOP
NAME DSNCSTOP(R)
ORDER DFHEAI,DFHEAI0,DSNCSTRT,DSNAA
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAI0)
INCLUDE DSNLOAD(DSNCSTRT)
ENTRY DSNCSTRT
NAME DSNCSTRT(R)
ORDER DFHEAI,DFHEAI0,DSNCUEXT,DSNAA
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAI0)
INCLUDE DSNLOAD(DSNCUEXT)
ENTRY DSNCUEXT
NAME DSNCUEXT(R)
ORDER DFHEAI,DFHEAI0,DSNCEDF1,DSNAA
MODE AMODE(31),RMODE(24)
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAI0)
INCLUDE DSNLOAD(DSNCEDF1)
ENTRY DSNCEDF1
NAME DSNCEDF1(R)
ORDER DFHEAI,DFHEAI0,DSNCEDON,DSNAA
MODE AMODE(31),RMODE(24)
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAI0)
INCLUDE DSNLOAD(DSNCEDON)
ENTRY DSNCEDON
NAME DSNCEDON(R)
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAI0)
INCLUDE DSNLOAD(DSNCSM31)
ENTRY DSNCSM31
ORDER DFHEAI,DFHEAI0,DSNCSM31,DSNAA
MODE AMODE(31),RMODE(24)
NAME DSNCSM31(R)
INCLUDE DLOADLIB(DFHEAI)
INCLUDE DLOADLIB(DFHEAI0)
INCLUDE DSNLOAD(DSNCSM21)
ENTRY DSNCSM21
ORDER DFHEAI,DFHEAI0,DSNCSM21,DSNAA
MODE AMODE(31),RMODE(24)
NAME DSNCSM21(R)
```

Figure 9 (Part 2 of 2). Link-editing Step from DB2 3.1 Version of DSNTIJSU

DSNTIJSU, as supplied and as shown in Figure 9 on page 21, refers to CICS170.LOADLIB. This should have been replaced with the CICS LOADLIB name you defined in field 5 of the DSNTIPV panel during DB2 installation. If you are using multiple levels of CICS (for example, CICS/ESA 3.2.1 and CICS/ESA 3.3), specify the latest level of CICS you installed (up to CICS/ESA 3.3). The DFHEAI and DFHEAI0 stubs are compatible across CICS releases. This also means that you do not have to repeat this link-edit if you install a new version of CICS.

2.3.2 Providing CICS Access to the Attachment Facility Modules

The CICS attachment facility modules are found in DSN310.SDSNLOAD or DSN230.DSNLOAD. Some of these modules are defined to CICS and are loaded by CICS. Other modules are not defined to CICS and are loaded by MVS. The DSN310.SDSNLOAD or DSN230.DSNLOAD library must therefore be included in both of the following:

- The STEPLIB or JOBLIB concatenation, for the CICS attachment facility modules which are not defined to CICS, and so have to be loaded by MVS.
- The DFHRPL concatenation, for the modules defined to CICS.

  The modules that are defined to CICS are:

  DSNCCOM0
  DSNCCOM1
  DSNCCOM2
  DSNCEXT1
  DSNCEXT2
  DSNCMSG0
  DSNCSTOP
  DSNCSTRT
  DSNCUEXT
  DSNCEDF1
  DSNCEDON
  DSNCSM31 (for use with CICS/ESA Version 3)
  DSNCSM21 (for use with CICS/MVS Version 2)

The RCT is loaded using MVS services, not by using CICS; so the library containing the RCT must be concatenated to the CICS STEPLIB or to the JOBLIB.

The RCT can be in any one of these:

- The DSN310.SDSNLOAD or DSN230.DSNLOAD library
- The CICS authorized load library containing DFHSIP
- Any other library that is defined to MVS as authorized program facility (APF)-authorized.

The library containing the RCT must be APF-authorized. This is required because the library containing DFHSIP (the CICS system initialization program) is included in the STEPLIB concatenation, and this library must be APF-authorized. The RCT itself is not authorized, but because it is loaded by MVS it is loaded from STEPLIB. All STEPLIB libraries need to be authorized so that CICS can be initialized.

Figure 8 on page 19 shows how DSN310.SDSNLOAD is used in the CICS start-up procedure. It also shows DSN310.RCT.LOAD, a separate APF-authorized library just for the RCT.

2.3.3 Defining the CICS Connection to DB2

The connection from CICS to DB2 is defined in the RCT, using macros provided by DB2. You can have several different RCTs defined in your installation, but a CICS region uses only one RCT at any one time. The RCTs are distinguished by a 1- or 2-character suffix that is specified when the RCT is generated.

Figure 10 on page 24 shows the RCT supplied for the sample applications. Note that in this example most of the parameters in the TYPE=INIT macro for the supplied RCT were allowed to take their default values. We specified a value only for SUFFIX. The most significant defaults are:

SUFFIX The default character is 0; so the name of the generated RCT is DSNCRCT0 if no SUFFIX is specified. The RCT definition in Figure 10 generates an RCT called DSNRCTZ.

SUBID This parameter identifies the name of the DB2 subsystem that CICS is to connect to using the CICS attachment facility. The default name is DSN. You should specify a SUBID if your DB2 subsystem name is not DSN.

The POOL and ENTRY thread definitions in Figure 10 do not specify a value for the AUTH parameter. The default value is AUTH=(USER,TERM,TXID), meaning that the authorization ID passed from CICS to DB2 is:

- USER—Passes the 3-character operator identifier (padded with 5 blanks) for the signed-on user, if the transaction using the thread is running at a terminal and the terminal user signed on (or the TERMINAL definition includes a USERID)
- TERM—Passes the terminal identifier if the transaction using the thread is running at a terminal and the terminal user is not signed on (and the TERMINAL definition does not include a USERID)
- TXID—Passes the transaction identifier if the transaction using the thread is not running at a terminal.

Refer to Chapter 5, “Defining the Resource Control Table” on page 75 and the *IBM DATABASE 2 Administration Guide* for your release of DB2 for a full description of the macros involved in RCT generation. Chapter 4, “Security” on page 61 explains the impact that your specification of the AUTH parameter has at execution time.

DSN310.SDSNSAMP(DSN8FRCT)

```
      EJECT
      DSNCRCT TYPE=INIT,SUFFIX=Z
      DSNCRCT TYPE=POOL,
          THRDM=3,THRDA=3,PLAN=DSN8CC0
*
*     DEFINE THE PLI SAMPLE TRANSACTION
*
      DSNCRCT TYPE=ENTRY,TXID=D8PS,THRDM=1,THRDA=1,PLAN=DSN8CP0
      DSNCRCT TYPE=ENTRY,TXID=D8PP,THRDM=1,THRDA=1,PLAN=DSN8CQ0
*
*     DEFINE THE COBOL SAMPLE TRANSACTION
*
      DSNCRCT TYPE=ENTRY,TXID=D8CS,THRDM=1,THRDA=1,PLAN=DSN8CC0
*
*     DEFINE THE PLI PHONE  TRANSACTION
*
      DSNCRCT TYPE=ENTRY,TXID=D8PT,THRDM=1,THRDA=1,PLAN=DSN8CH0
*          D8PU TRANSACTION USED INTERNALLY
      DSNCRCT TYPE=ENTRY,TXID=D8PU,THRDM=1,THRDA=1,PLAN=DSN8CH0
      DSNCRCT TYPE=FINAL
      END
```

Figure 10. RCT Definition Supplied with Sample Programs

2.3.4 Updating CICS Definitions and Tables

We list the modules that you must define to CICS as part of the CICS attachment facility installation process in section 2.3.2, “Providing CICS Access to the Attachment Facility Modules” on page 23. DB2 supplies sample definitions for these CICS attachment facility modules and their related transactions. With CICS/ESA Version 3, you can use either resource definition online (RDO) or the DFHCSDUP utility to define the programs and transactions to CICS.

The DSNTIJSU job automatically uses DFHCSDUP to add the required program and transaction definitions to the CSD. You control some of the definitions by the parameters you specify on the DSNTIPV installation panel. You must supply the CSD name (field 6), an RDO group name (field 7), tell CICS whether this is a new group (field 8), and give the RDO list name (field 9). You must ensure that the RDO group name (field 7) is unique, because the DSNTIJSU job either adds a new definition (if field 8 says yes) or deletes the existing group and then adds it (if field 8 says no). You must also be sure that this group is in the list specified in the GRPLIST parameter at CICS startup (field 9).

The CICS tables and RDO definitions that can need changes to support the CICS attachment facility include:

- Program definitions
- Transaction definitions
- Program list table for post-initialization (PLTPI)
- Program list table for shutdown (PLTSD)
- System initialization table (SIT)

The suffix of the PLTPI and PLTSD used by CICS are specified in the SIT parameters PLTPI and PLTSD.

2.3.4.1 Program Definitions

Figure 11 on page 26 shows the CEDA commands, or DFHCSDUP statements, needed to make the program definitions required to operate the CICS attachment facility and run the sample applications.

```
*
*CICS PROGRAM ADDITIONS NEEDED FOR THE ATTACHMENT FACILITY *
*
DEFINE PROGRAM(DSNCCOM0) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(DSNCCOM1) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(DSNCCOM2) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(DSNCEDF1) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(DSNCEDON) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(DSNCEXT1) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(DSNCEXT2) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(DSNCMSG0) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(DSNCSTOP) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(DSNCSTRT) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(DSNCUEXT) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(DSNCSM21) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(DSNCSM31) GROUP(DB2) LANGUAGE(ASSEMBLER) CEDF(NO)
*
*SAMPLE MAPS AND PROGRAMS FOR SAMPLE APPLICATION
*
DEFINE MAPSET(DSN8CCD) GROUP(DB2)
DEFINE MAPSET(DSN8CCG) GROUP(DB2)
DEFINE MAPSET(DSN8CPD) GROUP(DB2)
DEFINE MAPSET(DSN8CPE) GROUP(DB2)
DEFINE MAPSET(DSN8CPF) GROUP(DB2)
DEFINE MAPSET(DSN8CPG) GROUP(DB2)
DEFINE MAPSET(DSN8CPL) GROUP(DB2)
DEFINE MAPSET(DSN8CPN) GROUP(DB2)
DEFINE MAPSET(DSN8CPU) GROUP(DB2)
*
DEFINE PROGRAM(DSN8CC0) GROUP(DB2) LANGUAGE(COBOL)
DEFINE PROGRAM(DSN8CC1) GROUP(DB2) LANGUAGE(COBOL)
DEFINE PROGRAM(DSN8CC2) GROUP(DB2) LANGUAGE(COBOL)
DEFINE PROGRAM(DSN8CP0) GROUP(DB2) LANGUAGE(PLI)
DEFINE PROGRAM(DSN8CP1) GROUP(DB2) LANGUAGE(PLI)
DEFINE PROGRAM(DSN8CP2) GROUP(DB2) LANGUAGE(PLI)
DEFINE PROGRAM(DSN8CP3) GROUP(DB2) LANGUAGE(PLI)
DEFINE PROGRAM(DSN8CP6) GROUP(DB2) LANGUAGE(PLI)
DEFINE PROGRAM(DSN8CP7) GROUP(DB2) LANGUAGE(PLI)
DEFINE PROGRAM(DSN8CP8) GROUP(DB2) LANGUAGE(PLI)
DEFINE PROGRAM(DSNTIAC) GROUP(DB2) LANGUAGE(ASSEMBLER)
DEFINE PROGRAM(DSNTIA1) GROUP(DB2) LANGUAGE(ASSEMBLER)
*
*DEFINE THE INITIALIZATION AND TERMINATION PLTs
*
DEFINE PROGRAM(DFHPLTTI) GROUP(DB2) LANGUAGE(ASSEMBLER)
DEFINE PROGRAM(DFHPLTTT) GROUP(DB2) LANGUAGE(ASSEMBLER)
```

Figure 11. Sample Program Definitions

CICS Attachment Facility Modules: The *IBM DATABASE 2 Administration Guide* gives a list of the CICS attachment facility modules that must be defined to CICS. Note that these definitions are not contained in the PPT sample shipped with the DB2 sample applications (member DSN8FPPT in DSN310.SDSNSAMP or DSN230.DSNSAMP). The definitions are supplied only as sample DEFINE statements for DFHCSDUP in DSNTIJSU.

CICS Resource Manager Definitions: Use the CICS-supplied resource group DFHRMI (included in DFHLIST list) to specify the resource manager interface (RMI).

Sample Programs Shipped with DB2: The definitions of these programs are contained in member DSN8FPPT in DSN310.SDSNSAMP or DSN230.DSNSAMP.

PLTPI and PLTSD (Optional): If you want to start the CICS attachment facility automatically when CICS is started, we recommend that you use the PLTPI. In this case the PLTPI must be defined to CICS as an assembler program. Section 2.3.4.3, “Updating the PLTPI” on page 30 gives a description of this facility.

The PLTSD definition is required if the CICS attachment facility is to be automatically terminated when CICS is terminated. We strongly recommend that you implement this, since CICS cannot terminate normally if the CICS attachment facility is still active. CICS shutdown waits until the operator enters a DSNC STOP command.

If the CICS operator requests an immediate shutdown of CICS (with a CEMT P SHUT IMM command) while the CICS attachment facility is still active, CICS abends with a return code of A03. This means that the CICS main TCB still had a subtask-TCB (the CICS attachment facility) attached when trying to terminate.

Other Program Definition Considerations: You must define your own application programs to CICS. If you are using CICS/ESA 3.3 with storage protection active, you must ensure that the following programs are defined with EXECKEY(CICS).

DSNCCOM1
DSNCEXT1
DSNCEXT2
DSNCMSG0
DSNCSM31
DSNCSTOP
DSNCSTRT
DSNCUEXT

You can do this either by editing the DSNTIJSU job before you submit it (if installing for the first time), or by using the CICS CEDA transaction to alter the existing definitions.

Your own programs should be defined as EXECKEY(USER).

2.3.4.2 Transaction Definitions

Figure 12 on page 28 shows the CEDA commands, or DFHCSD statements, needed to make the transaction definitions required to operate the CICS attachment facility and run the sample applications.

```
DELETE GROUP(DB2)
ADD    GROUP(DB2) LIST(DSNLIST)
*
*CICS TRANSACTION DEFINITIONS FOR THE ATTACHMENT FACILITY
*
DEFINE TRANSACTION(-DIS) GROUP(DB2) PROGRAM(DSNCCOM1) TWASIZE(1200) TASKDATALOC(BELOW)
DEFINE TRANSACTION(-REC) GROUP(DB2) PROGRAM(DSNCCOM1) TWASIZE(1200) TASKDATALOC(BELOW)
DEFINE TRANSACTION(-STA) GROUP(DB2) PROGRAM(DSNCCOM1) TWASIZE(1200) TASKDATALOC(BELOW)
DEFINE TRANSACTION(-STO) GROUP(DB2) PROGRAM(DSNCCOM1) TWASIZE(1200) TASKDATALOC(BELOW)
DEFINE TRANSACTION(DISC) GROUP(DB2) PROGRAM(DSNCCOM1) TWASIZE(1200) TASKDATALOC(BELOW)
DEFINE TRANSACTION(DISP) GROUP(DB2) PROGRAM(DSNCCOM1) TWASIZE(1200) TASKDATALOC(BELOW)
DEFINE TRANSACTION(DSNC) GROUP(DB2) PROGRAM(DSNCCOM1) TWASIZE(1200) TASKDATALOC(BELOW)
DEFINE TRANSACTION(MODI) GROUP(DB2) PROGRAM(DSNCCOM1) TWASIZE(1200) TASKDATALOC(BELOW)
DEFINE TRANSACTION(STOP) GROUP(DB2) PROGRAM(DSNCCOM1) TWASIZE(1200) TASKDATALOC(BELOW)
DEFINE TRANSACTION(STRT) GROUP(DB2) PROGRAM(DSNCCOM1) TWASIZE(1200) TASKDATALOC(BELOW)
*
*CICS TRANSACTION ADDITIONS NEEDED FOR THE DB2 SAMPLE PROGRAMS
*
DEFINE TRANSACTION(D8CS) GROUP(DB2) PROGRAM(DSN8CC0)
DEFINE TRANSACTION(D8PP) GROUP(DB2) PROGRAM(DSN8CP6)
DEFINE TRANSACTION(D8PS) GROUP(DB2) PROGRAM(DSN8CP0)
DEFINE TRANSACTION(D8PT) GROUP(DB2) PROGRAM(DSN8CP3)
DEFINE TRANSACTION(D8PU) GROUP(DB2) PROGRAM(DSN8CP3)
*
```

Figure 12. Transaction Definitions for the CICS Attachment Facility

CICS Attachment Facility Transactions: You can enter both CICS attachment facility commands and DB2 subsystem commands from a CICS terminal, as long as you are properly authorized. The difference between CICS attachment facility commands and DB2 commands is that CICS attachment facility commands do ***not*** have **-** as their first character, while DB2 commands do.

The command verb entered is preceded by a CICS transaction ID (DSNC is the transaction ID used in the *IBM DATABASE 2 Administration Guide*). The DSNC transaction invokes the command processor program (DSNCCOM1). You can invoke DSNCCOM1 using transaction IDs other than DSNC. DSNCCOM1 scans the first two character strings of the terminal input, looking for a valid CICS attachment facility or DB2 command verb. This means that you can define other CICS transaction IDs, based on the first 4 characters of a CICS attachment facility or DB2 command, and associate these new transactions with DSNCCOM1, so that the command need not be preceded by the DSNC transaction ID. For example, instead of typing DSNC STOP to stop the CICS attachment facility, you can define a transaction called STOP and associate it with program DSNCCOM1. Then you simply type STOP to stop the CICS attachment facility. Figure 12 shows all the transaction definitions you need to make to use the CICS attachment facility and DB2 commands directly. DSNTIJSU makes these definitions for you if you are using CICS/ESA Version 3. The transaction definitions are:

- -DIS—Allows you to issue any DB2 display command against databases, threads, traces, or utilities.
- -REC—Allows you to issue the DB2 commands to reestablish dual bootstrap data sets (BSDS), or to recover any in-doubt threads.
- -STA—Allows you to start databases or traces. You cannot start DB2 using this command from CICS, because the CICS attachment facility cannot be active when DB2 is down.

- -STO—Allows you to stop databases or traces. You can even stop DB2 using this command; in that case the operator does not have to stop the CICS attachment facility manually to allow DB2 shutdown to complete. (This is different from stopping DB2 from DB2I.)
- DISC—Allows you to get manual control to disconnect threads and release resources being shared by normal transactions.
- DISP—Allows you to display the status of a plan, a transaction, or the connection statistics by entering

  ```
  DISP xxxx xxxx
  ```

  instead of

  ```
  DSNC DISP xxxx xxxx
  ```

  .
- MODI—Allows you to modify:
  - The ERRDEST entry in the RCT, or
  - The maximum active thread value associated with a given transaction or group. A group is a set of transaction IDs defined in the same RCT entry.
- STOP—Allows you to stop the CICS attachment facility.
- STRT—Allows you to start the CICS attachment facility.

You can make similar entries for other CICS attachment facility commands.

In summary, you must make the following transaction definitions to support the CICS attachment facility and DB2 commands:

- DSNC. You must define this transaction to CICS.
- Each CICS attachment facility command to be entered directly without the DSNC transaction code (first 4 characters of the command). These entries must be related to the DSNCCOM1 command processor program.
- Each DB2 command to be entered directly without the DSNC transaction code (first 4 characters of the command). These entries must be related to the DSNCCOM1 command processor program.

CICS Resource Manager Definitions: Use the CICS-supplied resource group DFHRMI (included in DFHLIST list) to specify the resource manager interface (RMI).

Transactions for the Sample Programs: The definition of these transactions is contained in the DSNTIJSU job.

Other Transaction Definition Considerations: You must, at some stage, make definitions for your own transactions that will use DB2. No special action is necessary to activate dynamic transaction backout. With RDO each transaction automatically activates dynamic transaction backout, and the DTB=NO option is not allowed.

If you are running CICS/ESA 3.2.1 or CICS/ESA 3.3 you must specify TASKDATALOC(BELOW) in your transaction definitions. This restriction is removed with CICS/ESA 4.1. If you are running CICS/ESA 3.3 you should specify TASKDATAKEY(USER) for the transaction definitions.

2.3.4.3 Updating the PLTPI

Figure 13 shows a sample of the entry you should add to your PLTPI to start the CICS attachment facility automatically following CICS initialization.

```
PLTTI    DFHPLT TYPE=INITIAL,SUFFIX=TI
                        ...
         DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
                        ...
*
*        DEFINE THE ATTACHMENT FACILITY PROGRAM
*
                        ...
         DFHPLT TYPE=ENTRY,PROGRAM=DSNCCOM0
                        ...
                        ...

         DFHPLT TYPE=FINAL
         END
```

Figure 13. Sample CICS/ESA Version 3 PLTPI Entry for DSNCCOM0

Note: If you are running DB2 2.3 and have not applied APAR PN29087, then you should use DSNCCOM1 rather than DSNCCOM0 in your PLTPI. With CICS/ESA Version 3 you must place the entry for DSNCCOM0 after the entry for DFHDELIM. Using DSNCCOM0, as shown in Figure 13, starts the CICS attachment facility using the RCT specified in the PARM= statement of your CICS DFHSIP step, or the default DSNCRCT0 if no RCT was specified. For example, specify

```
//CICS  EXEC PGM=DFHSIP,PARM=('DSNCRCT4,SIT=61,SYSIN')
```

in your CICS job control language (JCL) to start CICS using DFHSIT61 as the SIT. The SIT should contain a statement specifying which PLTPI to use (PLTPI=TI to use the sample shown in Figure 13). When DSNCCOM0 is started by the PLTPI it uses DSNCRCT4 (the RCT generated with SUFFIX=4 in the TYPE=INIT statement). Other CICS initialization parameters are read from the SYSIN ddname. You could specify the PLT names in SYSIN rather than in the SIT, as shown in Figure 8 on page 19.

If you do not want to start the CICS attachment facility automatically, use the `DSNC STRT x` command from a CICS terminal to start the connection (where x is the suffix of the RCT you want to use).

2.3.4.4 Updating the PLTSD

Figure 14 on page 31 shows a sample of the entry required in the PLTSD to terminate the connection when CICS is terminated normally.

```
PLTTT    DFHPLT TYPE=INITIAL,SUFFIX=TT
                        ...
                        ...
*
*        DEFINE THE ATTACHMENT FACILITY PROGRAM
*
         DFHPLT TYPE=ENTRY,PROGRAM=DSNCCOM2
                        ...
                        ...
         DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
                        ...
                        ...
         DFHPLT TYPE=FINAL
         END
```

Figure 14. Sample CICS/ESA Version 3 PLTSD Entry for DSNCCOM2

DSNCCOM2 must run during the first quiesce stage of termination; so it must be placed before the entry for DFHDELIM.

Note: You must define transaction DSNC to CICS if you want to use a PLTSD entry to stop the CICS attachment facility. This is because the program DSNCCOM2 starts this transaction ID when invoked this way.

We recommend that you use the PLTSD to stop the CICS attachment facility, thereby avoiding operator intervention.

2.3.4.5 SQL Statements in PLTPI Programs

In CICS/ESA Version 3, programs running under the CPLT transaction cannot make SQL calls. To start an SQL application from the PLTPI, the program you name in the PLTPI should start a separate task using the EXEC CICS START command with the INTERVAL(x) option. The INTERVAL option ensures that the CICS attachment facility starts before the new task attempts to make any SQL calls. An alternative would be to have the new task check to see whether the CICS attachment facility is enabled before making any SQL calls. As an example, if the PLTPI shown in Figure 23 on page 42 were for a CICS/ESA Version 3 region, PLTPROG0 could not include any SQL calls. It could, however, include an EXEC CICS START command for another transaction, and the program associated with that transaction could include SQL calls.

2.3.4.6 Updating the System Initialization Table

The changes required to the SIT to support the CICS attachment facility are:

- DSA sizes

  Each CICS thread to DB2 requires an MVS TCB for its operation. MVS storage for each thread is dynamically acquired from the storage area left by the difference between the CICS region size and the sum of the DSA sizes coded in the SIT.

  You must estimate the amount of storage required. Take into account the number of threads defined, and ensure that the storage available is large enough to accommodate the CICS attachment facility control blocks and tables. The *IBM DATABASE 2 Administration Guide* for your release of DB2 contains storage estimates to help you complete this task.

Execution Diagnostic Facility (EDF) information is stored in the EDSA. Refer to the *IBM DATABASE 2 Administration Guide* for your release of DB2 to calculate the proper space requirements for your environment.

- PLTPI and PLTSD

  Include entries for the PLTPI and PLTSD if you want to use these to automatically start and stop the CICS attachment facility.

- GRPLIST

  Ensure that the DB2 and DFHRMI groups are in the RDO list of groups that CICS uses for initialization to allow access to the CICS attachment facility transaction codes and programs.

2.3.5 Coordinating CICS and DB2 Security

Security in a CICS-DB2 environment is discussed in Chapter 4, “Security” on page 61. This discussion of security is limited to the authorizations required to run the IVPs, issue CICS attachment facility commands, and issue DB2 commands.

Authorizations needed to run the IVPs are:

- RACF authorization for the CICS connection to DB2, if RACF is present.
- BIND or BINDADD privileges for the installer to bind the sample application plans. The installer also needs the table privileges used in the sample application, if these tables have not been made PUBLIC using a DB2 GRANT command.
- EXECUTE privilege on the sample application plans for the CICS users who will enter the CICS transactions.

  You should use the DB2 GRANT command to give these CICS users authority use the sample application plans. You need to give GRANT authority according to the AUTH parameter in the DSNCRCT TYPE=ENTRY. No AUTH parameter is defined in the default DSNCRCT provided in DSN8FRCT member of DSN230.DSNSAMP or DSN310.SDSNSAMP (refer to Figure 10 on page 24). This means that the authorization ID used by the CICS attachment facility is:

  – USER—The 3-character operator identifier (padded with 5 blanks) for the signed on user, if the transaction using the thread is running at a terminal and the terminal user signed on, or a USERID is specified in the TERMINAL definition

  – TERM—The terminal identifier if the transaction using the thread is running at a terminal and the terminal user is not signed on, and a USERID is not specified in the TERMINAL definition

  – TXID—If the transaction using the thread is not running at a terminal.

  After you verify that installation is successfully completed, you can control access to CICS and DB2 resources using the facilities described in Chapter 4, “Security” on page 61.

The CICS users authorized to issue CICS attachment facility commands (for example, `DSNC STOP`) do not need any authority within DB2, because these commands do not go to DB2.

You should use GRANT to give the appropriate authority to the authorization IDs of the CICS users authorized to issue DB2 commands (for example, DSNC -DISPLAY DATABASE (*)). The authorization ID passed to DB2 is defined by the AUTH parameter in the DSNCRCT TYPE=COMD. No AUTH parameter is defined in the default DSNCRCT provided in the DSN8FRCT member of DSN310.DSNSAMP (refer to Figure 10 on page 24). This means that the authorization ID used by the attachment facility is:

- USER—The 3-character operator identifier (padded with 5 blanks) for the signed on user, if the terminal user signed on, or a USERID is specified in the TERMINAL definition
- TERM—The terminal identifier if the terminal user is not signed on, and a USERID is not specified in the TERMINAL definition.

2.3.6 Example of Security Coordinated between CICS and DB2

Follow these steps to check the coordinated security between CICS and DB2:

1. REVOKE all authority on CICS plans provided by the IVPs. This is because these plans were made PUBLIC using the DB2 GRANT command. You can use the following statement, if your authorization ID has SYSOPER authority:

   ```
   REVOKE EXECUTE ON PLAN DSN8CCP0,
          DSN8CCQ0, DSN8CCC0, DSN8CH0 FROM PUBLIC;
   ```

2. Use the DB2 GRANT command to give EXECUTE authority on CICS plans to the specific user you want to sign on (8-character user ID). Then, use the GRANT command to give the same user ID the DISPLAY privilege. You can use the following statements:

   ```
   GRANT  EXECUTE ON PLAN DSN8CCP0,
          DSN8CCQ0, DSN8CCC0, DSN8CH0 TO userid ;
   GRANT DISPLAY TO userid ;
   ```

3. If you are not using CICS/ESA 3.3 (or later) with RACF 1.9 (or later), prepare a CICS sign-on table (SNT), using the 8-character USERID parameter where appropriate. Figure 15 gives an example of how to do this.

```
SNT      TITLE 'DFHSNTDB'
*
         DFHSNT TYPE=INITIAL
*
         DFHSNT TYPE=ENTRY,                                          X
               OPIDENT=STA,                                          X
               OPPRTY=255,                                           X
               TIMEOUT=60,                                           X
               USERID=STABERG
*
         DFHSNT TYPE=(ENTRY,DEFAULT),                                X
               TIMEOUT=10
*
         DFHSNT TYPE=FINAL
         END   ,
```

Figure 15. Sample SNT Using the 8-Character USERID

With CICS/ESA 3.3 and RACF 1.9 there is no need to create an SNT. Users must be defined to RACF.

4. Prepare an RCT with the AUTH parameter. Figure 16 on page 34 gives an example of how to do this. In Figure 16 on page 34 the DSNCRCT TYPE=COMD entry sets the AUTH parameter to AUTH=(USERID,*,*). Default values are provided for the command thread. You do not need to include a definition for the command thread in your RCT if these defaults meet your needs. The AUTH parameter was added to every DSNCRCT TYPE=ENTRY macro. Its value specifies using the 8-character USERID as the authorization ID.

```
         DSNCRCT TYPE=INIT,SUFFIX=2,SUBID=DSN3
*
         DSNCRCT TYPE=COMD,DPMODE=HIGH,TXID=DSNC,                      X
               AUTH=(USERID,*,*),                                      X
               ROLBE=NO,                                               X
               THRDM=1,                                                X
               THRDA=1,                                                X
               THRDS=1,                                                X
               TWAIT=POOL
*
         DSNCRCT TYPE=POOL,                                            X
               THRDM=3,THRDA=3,PLAN=DSN8CCC0
*
*        DEFINE THE PLI SAMPLE TRANSACTION
*
         DSNCRCT TYPE=ENTRY,TXID=D8PS,THRDM=1,THRDA=1,PLAN=DSN8CCP0,   X
               AUTH=(USERID,*,*)
         DSNCRCT TYPE=ENTRY,TXID=D8PP,THRDM=1,THRDA=1,PLAN=DSN8CCQ0,   X
               AUTH=(USERID,*,*)
*
*        DEFINE THE COBOL SAMPLE TRANSACTION
*
*        DSNCRCT TYPE=ENTRY,TXID=D8CS,THRDM=1,THRDA=1,PLAN=DSN8CCC0
         DSNCRCT TYPE=ENTRY,TXID=D8CS,THRDM=5,THRDA=1,PLAN=DSN8CCC0,   X
               AUTH=(USERID,*,*)
*
*        DEFINE THE PLI PHONE  TRANSACTION
*
         DSNCRCT TYPE=ENTRY,TXID=D8PT,THRDM=1,THRDA=1,PLAN=DSN8CCH0,   X
               AUTH=(USERID,*,*)
*                D8PU TRANSACTION USED INTERNALLY
         DSNCRCT TYPE=ENTRY,TXID=D8PU,THRDM=1,THRDA=1,PLAN=DSN8CCH0,   X
               AUTH=(USERID,*,*)
         DSNCRCT TYPE=FINAL
         END
```

Figure 16. Sample RCT Using AUTH=(USERID,,*)*

5. After starting CICS and the CICS attachment facility, sign on to CICS using the CESN transaction.

2.3.7 Installation Verification Procedure

Verifying the DB2 installation is discussed in the *IBM DATABASE 2 Administration Guide*.

Phase 5 of the installation verification procedure tests the CICS attachment facility. Job streams are supplied in DSN230.DSNSAMP or DSN310.SDSNSAMP to install the sample application transactions:

- COBOL (member DSNTEJ5C)
- PL/I (member DSNTEJ5P).

We recommend using these job streams to test and gain experience in operating the CICS attachment facility.

Before you start, make sure that the previous phases of the IVP (except Phase 4) were run, and that the DB2 objects created in these phases (including DCLGENs) were retained. If they were not retained, the DB2 administrator must rerun selected parts of the IVP to provide the objects required for testing the CICS attachment facility.

Once you select the languages and applications to use for the IVPs, you should change the library names used in the sample job steps to conform to your installation standards. The IVP steps are distributed as a single job stream, but you may find it more convenient to run each step separately because of the volume of output generated.

The *IBM DATABASE 2 Administration Guide* explains how to prepare the IVP applications using members DSNTEJ5C and DSNTEJ5P (in DSN230.DSNSAMP or DSN310.SDSNSAMP). The DSNH CLIST prepares the COBOL sample programs. You may find it difficult to tailor this to your specific environment. Figure 17 on page 36 is a sample procedure using standard JCL, which you can use to prepare CICS-DB2 COBOL programs.

Note that the sample application programs are link-edited into the DSN310.RUNLIB.LOAD library. You can move the sample application programs to another load library to conform to installation standards, or DSN310.RUNLIB.LOAD must be part of the concatenation of DFHRPL (see Figure 8 on page 19).

```
//TESTCOB JOB  CLASS=A,REGION=4500K,
//             USER=SYSADM,PASSWORD=SYSADM,MSGCLASS=H
//*********************************************************************
//DB2     EXEC PGM=DSNHPC,PARM='HOST(COBOL),XREF,SOURCE,FLAG(I),APOST'
//STEPLIB  DD  DSN=USER.CICS.TESTLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSTERM  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//DBRMLIB  DD  DSN=USER.DBRMLIB(TESTCOB),DISP=SHR
//SYSCIN   DD  DSN=&&DBSOURCE,DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//             SPACE=(TRK,(10,1))
//SYSIN    DD  *
//*PLACE YOUR COBOL PROGRAM HERE
//*********************************************************************
//*        CICS COMMAND LEVEL TRANSLATOR                              |
//*********************************************************************
//TRN     EXEC PGM=DFHECP1$,COND=(4,LT,DB2),
//             PARM='NOSOURCE,APOST,COBOL2,FLAG(I),VBREF,NOOPT'
//STEPLIB  DD  DSN=CICS330.SDFHLOAD,DISP=SHR
//SYSPUNCH DD  DSN=&&SOURCE,
//             DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//             SPACE=(TRK,(5,5))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DSN=&&DBSOURCE,DISP=(OLD,PASS)
//*********************************************************************
//*        COMPILE CICS PROGRAM                                       |
//*********************************************************************
//COB     EXEC PGM=IGYCRCTL,COND=((4,LT,DB2),(5,LT,TRN)),
// PARM='NOTRUNC,OBJECT,LIB,APOST,RES,RENT,LIST,DATA(31),NODYN,NOOPT'
//STEPLIB  DD  DSN=SYS1.COB2LIB,DISP=SHR
//SYSLIB   DD  DSN=SYS1.COB2CICS,DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT5   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSLIN   DD  DSN=&&OBJMOD(TESTC01),DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//             SPACE=(TRK,(5,2,1))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DSN=&&SOURCE,DISP=(OLD,DELETE)
//*********************************************************************
//*        LINKEDIT CICS PROGRAM                                      |
//*********************************************************************
//LNKEDT  EXEC PGM=IEWL,PARM='LIST,XREF,LET,AMODE=31,RMODE=31',
//             COND=((4,LT,DB2),(5,LT,TRN),(5,LT,COB))
//SYSLIB   DD  DSN=CICS330.SDFHLOAD,DISP=SHR
//         DD  DSN=SYS1.COB2CICS,DISP=SHR
//         DD  DSN=SYS1.COB2LIB,DISP=SHR
//         DD  DSN=DSN310.SDSNLOAD,DISP=SHR
//OBJLIB   DD  DSN=&&OBJMOD,DISP=(OLD,DELETE)
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT DD  SYSOUT=*
//SYSLMOD  DD  DSN=DSN310.RUNLIB.LOAD,DISP=SHR
//SYSLIN   DD  *
 INCLUDE SYSLIB(DFHECI)
 INCLUDE OBJLIB(TESTCOB)
 INCLUDE SYSLIB(DSNCLI)
 NAME TESTCOB(R)
```

Figure 17. Sample Procedure to Prepare CICS-DB2 COBOL Programs

Figure 18 on page 37 shows one of the steps of the BIND process for the COBOL phone application. Basic mapping support (BMS) map preparation and program translation steps are not shown because their operation is unchanged in a CICS-DB2 environment. Note that if the CICS translate step runs before the

DB2 precompiler, the EXEC SQL commands are flagged with a warning message, and the translator returns a condition code of 4.

The BIND process in Figure 18 results in the following:

- The BIND statement creates a plan named DSN8CCC0 from three database request modules (DBRMs), DSN8CC0, DSN8CC1, and DSN8CC2.
- The binder, STSYSAD in this case, must have all the required privileges on the tables and views, plus the BINDADD privilege. As a consequence of being the binder, STSYSAD gets EXECUTE authority with the GRANT option for the plan DSN8CCC0.
- Unqualified view or table names are qualified to the binder's ID, STSYSAD.
- ACT(REP) specifies that any existing application plan called DSN8CCC0 is to be replaced.
- The isolation is established at CURSOR STABILITY level. ***This should always be specified***, unless you need REPEATABLE READ.
- Validation of DB2 objects takes place at BIND time. From a performance viewpoint, ***this should always be specified***, although the default is VALIDATE(RUN).
- RETAIN means that the previous authorizations for the plan are not to be revoked. Note that the sample jobs in DSN310.SDSNSAMP do ***not*** specify RETAIN at BIND time, and RETAIN is not the default value. From an operational viewpoint, you should specify RETAIN. The sample jobs in DSN310.SDSNSAMP use packages, which simplifies the operation.

```
//STSYSADI JOB ,'STSYSAD',MSGCLASS=T,MSGLEVEL=(1,1),
//             NOTIFY=STSYSAD,REGION=3333K,USER=STSYSAD
//*
//*   BIND THE PROGRAM
//BIND     EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//DBRMLIB  DD DISP=SHR,
//            DSN=DSN310.DBRMLIB.DATA
//SYSUDUMP DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN  DD *
DSN SYSTEM(DSN)
BIND PLAN(DSN8CCC0) MEMBER(DSN8CC0,DSN8CC1,DSN8CC2) ACT(REP) ISO(CS) -
VALIDATE(BIND) -
RETAIN
END
//*
```

Figure 18. Example of the BIND Process Used for the Phone Application

2.3.7.1 Authorizing CICS Users to Execute the Plan

CICS terminal users have to be authorized before using a plan. As part of the IVP, EXECUTE privilege was granted to PUBLIC. This was done using the following SQL statement:

```
GRANT EXECUTE ON PLAN DSN8CCC0 TO PUBLIC
```

The plan was bound by STSYSAD; so this user ID is used to grant the execution privileges.

2.3.7.2 Create Synonym

You may need to create synonyms when using unqualified names for DB2 tables or views. In addition, you have to consider using unqualified names with static or dynamic SQL.

See Chapter 4, “Security” on page 61 for the implications of using unqualified names and static or dynamic SQL. This applies to both CREATE and BIND procedures.

2.3.7.3 Running the IVP CICS Applications

The *IBM DATABASE 2 Administration Guide* describes how to start and run an application in a CICS environment.

2.3.8 Using VS COBOL II or COBOL/370

There are no special considerations for using VS COBOL II with DB2. There are, however, some considerations for using VS COBOL II with CICS. These are listed in the section entitled “VS COBOL II considerations” in the *CICS/ESA Application Programming Guide*.

When you prepare your program, you should specify COB2 for the HOST DB2 precompiler parameter. The CICS considerations mean that if you prepare your CICS COBOL programs using the the DSNH CLIST, you should use CICSOPT(ANSI85), or CICSOPT(COBOL2) if you want to use the COMPR2 compiler option, as shown in Figure 19.

```
                          DB2 PROGRAM PREPARATION              SSID: DSN
COMMAND ===>

Enter the following:
 1  INPUT DATA SET NAME .... ===>
 2  DATA SET NAME QUALIFIER  ===> TEMP       (For building data set names)
 3  PREPARATION ENVIRONMENT  ===> FOREGROUND (FOREGROUND, BACKGROUND, EDITJCL)
 4  RUN TIME ENVIRONMENT ... ===> CICS   (TSO, CAF, CICS, IMS)
 5  OTHER DSNH OPTIONS ..... ===> CICSOPT(COBOL2)
                                          (Optional DSNH keywords)
Select functions:           Display panel?       Perform function?
 6  CHANGE DEFAULTS ........ ===> N (Y/N)
 7  PL/I MACRO PHASE ....... ===> N (Y/N)         ===> N (Y/N)
 8  PRECOMPILE ............. ===> Y (Y/N)         ===> Y (Y/N)
 9  CICS COMMAND TRANSLATION                      ===> N (Y/N)
10  BIND PACKAGE ........... ===> Y (Y/N)         ===> Y (Y/N)
11  BIND PLAN .............. ===> Y (Y/N)         ===> Y (Y/N)
12  COMPILE OR ASSEMBLE .... ===> Y (Y/N)         ===> Y (Y/N)
13  PRELINK ................ ===> N (Y/N)         ===> N (Y/N)
14  LINK ................... ===> N (Y/N)         ===> Y (Y/N)
15  RUN .................... ===> N (Y/N)         ===> Y (Y/N)

PRESS:  ENTER to process    END to save and exit     HELP for more information
```

Figure 19. DB2I Program Preparation Panel for CICS

If you create your own procedure for preparing CICS-DB2 applications, you should still specify COBOL2 (or ANSI85) as one of the options for the CICS translator.

2.4 CICS Attachment Facility Installation (CICS/ESA 4.1 Onward)

If you want to connect to DB2 from CICS/ESA 4.1, you must use the CICS attachment facility shipped with CICS/ESA 4.1, not the CICS attachment facility shipped with DB2.

This section assumes that you are migrating from an earlier version of CICS/ESA. If you are installing the CICS attachment facility for the first time, you must read section 2.3, "CICS Attachment Facility Installation (CICS/ESA Version 3)" on page 19 as well as this section.

The new CICS attachment facility is easier to install than the older version. However, if you are migrating from the older version of the CICS attachment facility, you have to take some extra steps to ensure that you are using the correct version of the CICS attachment facility.

2.4.1 Installing the CICS Attachment Facility

The CICS attachment facility is shipped as a feature of CICS/ESA Version 4. The FMID is JCI4106 for CICS/ESA 4.1.

Two of the CICS/ESA 4.1 installation jobs install the CICS attachment facility. DFHINST5, as part of its tasks, receives the attachment facility, while DFHINST6 does the SMP/E APPLY and ACCEPT processing. Do not run the DB2 installation job DSNTIJSU.

In the new CICS/ESA 4.1 version of the CICS attachment facility:

- The CICS attachment facility modules are shipped with the CICS stubs already linked to them.
- CICS provides the program and transaction definitions in a predefined group called DFHDB2.

2.4.1.1 Program Definitions

You do not need to define the CICS attachment facility modules in the CICS system definition file (CSD), because this is done during SMP/E installation steps.

If, for any reason, you ran the DB2 installation job DSNTIJSU, you must check for old CICS attachment facility program definitions in your CSD.

If you migrated completely to CICS/ESA 4.1, delete the following program definitions. If you are still running another level of CICS alongside CICS/ESA 4.1, ensure that the group containing the following program definitions is not included in any of the RDO lists used by the CICS/ESA 4.1 regions, and that the group is not installed in the CICS/ESA 4.1 region (especially if you are sharing one CSD between different levels of CICS).

```
DSNCCOM0
DSNCUEXT
DSNCCOM1
DSNCCOM2
DSNCEDF1
DSNCEDON
DSNCEXT1
DSNCEXT2
```

DSNCMSG0
DSNCSTOP
DSNCSTRT

Notice that *DSNCxxxx* modules belong to the old CICS attachment facility. The new attachment modules are named *DSN2xxxx*. Almost all of the CICS attachment facility modules have been renamed. If you have any application programs that reference CICS attachment facility module names, you must change your programs to use the new program names (shown in Figure 28 on page 46) and recompile them. With this exception, your own application programs can continue to run unchanged with the new CICS attachment facility. This is because the name of the attachment facility stub, DSNCLI, did not change and does not need to be relinked to your application programs. The name of the dynamic user exit program, DSNCUEXT, also stayed the same.

2.4.1.2 Transaction Definitions

As with program definitions, if you ran DSNTIJSU for any reason you must check for old CICS attachment facility transaction definitions in your CSD. You must look for DSNC or any other transaction that executes program DSNCCOM0, DSNCCOM1, or DSNCCOM2.

You may wish to alter these transactions to execute the new CICS attachment facility equivalents, or make new copies of the transaction definitions that execute the new CICS attachment facility modules. If you are running a mixed environment (multiple levels of CICS, sharing a CSD between levels), you must be very careful to ensure that your transactions invoke the correct CICS attachment facility modules for that level of CICS.

2.4.2 Virtual Storage Constraint Relief

The CICS attachment facility modules and most of their associated control block storage now resides above the 16M byte line. This reduces your virtual storage constraints. You can now specify TASKDATALOC(ANY) for your application programs that use SQL. Under the old CICS attachment facility shipped with DB2, CICS-DB2 application programs are restricted to TASKDATALOC(BELOW).

2.4.3 Preparing DB2

To prevent accidental use of the old CICS attachment facility with CICS/ESA 4.1, and to update the installation panels with the new version of CICS, there are prerequisite APARs that you must apply to DB2 3.1 and DB2 2.3. They are:

- PN52110—CICS attachment facility preconditioning
- PN52129—installation preconditioning

APAR PN52110 causes the message shown in Figure 20 to be displayed during CICS/ESA 4.1 initialization if you try to use the old CICS attachment facility.

```
DSNC057I csect-name CICS RELEASE release IS TOO HIGH FOR
THIS ATTACHMENT FACILITY
```

Figure 20. CICS/ESA 4.1 Initialization Message if Using the Old CICS Attachment Facility

2.4.4 Providing CICS Access to the Attachment Facility Modules

The STEPLIB statement of your CICS initialization JCL must include the library that contains your RCT. You do not need to include your DB2 load library (DSN230.DSNLOAD or DSN310.SDSNLOAD) in the STEPLIB statement to run the CICS attachment facility. If you choose to include your DB2 load library in the STEPLIB, it must be concatenated after CICS410.SDFHAUTH.

There should be no DB2 product libraries in the DFHRPL concatenation because the CICS attachment facility modules that must be used are found in CICS410.SDFHLOAD. Figure 21 shows part of a sample CICS initialization job that includes the STEPLIB and the DFHRPL statements.

```
//CICS41J  JOB  CLASS=5,MSGLEVEL=(1,1),MSGCLASS=Z,TIME=1440
//ONLINE   EXEC PGM=DFHSIP,TIME=1,REGION=7500K,DPRTY=(15,9),
//              PARM=('SYSIN')
//STEPLIB  DD  DSN=USER.CICS.SIT,DISP=SHR   CICS SIT TABLES
//         DD  DSN=USER.CICS.RCTLIB,DISP=SHR        DB2 RCT
//         DD  DSN=CICS410.SDFHAUTH,DISP=SHR
//         DD  DSN=DSN310.SDSNLOAD,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2CICS,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2LIB,DISP=SHR
//         DD  DSN=SYS2.LINKLIB,DISP=SHR
//DFHRPL   DD  DSN=USER.CICS.TABLES,DISP=SHR    CICS TABLES
//         DD  DSN=USER.CICS.RUNLIB,DISP=SHR    CICS APPLIC
//         DD  DSN=CICS410.SDFHLOAD,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2CICS,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2LIB,DISP=SHR
//         DD  DSN=SYS2.LINKLIB,DISP=SHR
```

Figure 21. Example of a CICS/ESA 4.1 Initialization Job

You can no longer specify your default RCT by using DSNCRCTx in the PARM parameter for the DFHSIP job. The DSNCRCTx initialization parameter has been replaced by the INITPARM option in the SIT (refer to the *CICS/ESA System Definition Guide* for CICS/ESA 4.1 for full information on the INITPARM SIT option). If you do not wish to specify the default RCT in the SIT, you can specify INITPARM in the PARM parameter for the DFHSIP job, or in the SYSIN dataset. The default resource control table suffix is now 00.

Section 2.4.5.3, “Updating the System Initialization Table” on page 42 gives a description of how to update your SIT to use the INITPARM option.

2.4.5 Updating CICS Definitions and Tables

You may need to change your PLTPI, PLTSD, and SIT to support the new CICS attachment facility.

2.4.5.1 Updates to Your PLTPI and PLTSD

You must delete any reference to DSNCCOM0, DSNCCOM1, or DSNCCOM2 in your PLTs. To start the CICS attachment facility, use DSN2COM0 in your PLTPI. To shut down the CICS attachment facility, use DSN2COM2 in your PLTSD. Figure 22 on page 42 gives an example of a CICS/ESA 4.1 PLTPI entry for DSN2COM0.

```
PLT4I    DFHPLT TYPE=INITIAL,SUFFIX=4I
                        ...
         DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
                        ...
*
*        DEFINE THE ATTACHMENT FACILITY PROGRAM
*
                        ...
         DFHPLT TYPE=ENTRY,PROGRAM=DSN2COM0
         DFHPLT TYPE=ENTRY,PROGRAM=PLTPROG0
                        ...
                        ...

         DFHPLT TYPE=FINAL
         END
```

Figure 22. Sample CICS/ESA 4.1 PLTPI Entry for DSN2COM0

Figure 23 gives an example of a CICS/ESA 4.1 PLTSD entry for DSN2COM2.

```
PLT4T    DFHPLT TYPE=INITIAL,SUFFIX=4T
                        ...
                        ...
*
*        DEFINE THE ATTACHMENT FACILITY PROGRAM
*
                        ...
         DFHPLT TYPE=ENTRY,PROGRAM=DSN2COM2
                        ...
         DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
                        ...

         DFHPLT TYPE=FINAL
         END
```

Figure 23. Sample CICS/ESA 4.1 PLTSD Entry for DSN2COM2

Shutting down the attachment facility during CICS initialization is not supported. Therefore you must not put DSN2COM2 in your PLTPI, even if DSN2COM0 is already there. DSN2COM2 is only supported in the PLTSD.

2.4.5.2 SQL Statements in PLTPI Programs

In CICS/ESA 4.1, programs running under the CPLT transaction can make SQL calls. If the PLTPI shown in Figure 23 were for a CICS/ESA 4.1 region, PLTPROG0 could include SQL calls.

2.4.5.3 Updating the System Initialization Table

The changes required to the SIT to support the CICS attachment facility are:

- DSALIM

  Each CICS thread to DB2 requires an MVS TCB for its operation. MVS storage for each thread is dynamically acquired from the storage left by the difference between the CICS region size and the DSALIM size coded in the

SIT. You must take into account the number of threads defined, and be sure that this difference is large enough to accommodate the CICS attachment facility control blocks.

EDF information is stored in EDSA. Refer to the *IBM DATABASE 2 Administration Guide* to calculate the proper space requirements for your environment.

- PLTPI and PLTSD

  Include entries for the PLTPI and PLTSD if you want to use these to automatically start and stop the CICS attachment facility.

- GRPLIST

  Ensure that the DFHDB2 and DFHRMI groups are in the RDO list of groups that CICS uses for initialization to allow access to the CICS attachment facility transaction codes and programs.

- INITPARM

  Set up the INITPARM option for RCT selection.

You can use either the INITPARM option at CICS initialization or the DSNC STRT command to select the RCT and the DB2 subsystem ID you want to use. Figure 24 shows the syntax of the INITPARM parameter.

```
INITPARM=(DSN2STRT='rct-suffix,ssid',other-initparms-->)
```

Figure 24. INITPARM Syntax

For example, if you say INITPARM=(DSN2STRT='aa,bbbb'), you use the resource control table named DSN2CTaa and you connect to DB2 subsystem bbbb, regardless of what subsystem is specified in DSN2CTaa.

Other examples are shown in Figure 25

```
INITPARM=(DSN2STRT='22')      RCT is DSN2CT22, no SSID
                              override

INITPARM=(DSN2STRT=',DSNA')   Use the default RCT, connect to
                              DSNA (In this example the
                              comma is used because the SSID
                              is a positional parameter, and
                              the comma indicates that no
                              RCT suffix is specified.)

INITPARM=(DSN2STRT='5,DSN',myparm='x')
                              RCT is DSN2CT5, SSID is DSN
                              (this example includes an
                              INITPARM for another CICS
                              application)
```

Figure 25. Examples of INITPARM Statements

The INITPARM contains single quotation marks; so you must double the single quotation marks to imbed them in the PARM= parameter, as shown in Figure 26 on page 44.

```
//ONLINE EXEC PGM=DFHSIP,TIME=1,REGION=7500K,DPRTY=(15,9),
//            PARM='SYSIN,INITPARM=(DSN2STRT=''41'')
```

Figure 26. INITPARM As a Parameter of DFHSIP

2.4.6 Defining the CICS/ESA 4.1 Connection to DB2

If you created an RCT using the older CICS attachment facility, you must reassemble it. You cannot use your old RCT with CICS/ESA 4.1 or later versions of CICS/ESA.

The RCT load module was renamed from DSNCRCTx to DSN2CTxx to allow for a 2-byte suffix. The macro that generates the RCT is still named DSNCRCT; so any current RCT source you have need not be recoded. However, you must use the DSNCRCT macro that is shipped in the CICS/ESA 4.1 macro library, CICS410.SDFHMAC, instead of the one in your DB2 macro library (DSN230.DSNMACS or DSN310.SDSNMACS). Thus the SYSLIB DD statement in the assemble step of your JCL to prepare the RCT must include the CICS410.SDFHMAC library, and it must not contain the DB2 macro library.

You can use the CICS procedure DFHAUPLE to assemble and link edit your resource control table. Figure 27 shows an example of the JCL you can use to assemble your RCT.

```
//RCT      EXEC DFHAUPLE,
// PARM.LNKEDT='AMODE=24,RMODE=24,LIST,XREF,LET,NCAL'
//ASSEM.SYSUT1 DD DSN=USER.RCT.SOURCE(DSN2RCT1),DISP=SHR
//LNKEDT.SYSLMOD DD DSN=USER.CICS.RCTLIB,DISP=SHR
```

Figure 27. Sample CICS/ESA 4.1 JCL for Resource Control Table Assembly

2.4.6.1 Updates to the Trace ID

The CICS attachment facility trace ID ranges changed from 1-199 to 256-511. If you explicitly coded the PLNXTR1, PLNXTR2, or TRACEID options in your RCT, you must change these options to use trace IDs within the new range. The default trace IDs for the old attachment facility are 192, 193, and 194 (X'C0', X'C1', and X'C2'). The corresponding default trace IDs for the new CICS attachment facility are 448, 449,and 450 (X'C0', X'1C1', and X'1C2').

2.4.7 Effect of Changed Module Names

You must change any application programs that link to DSNCCOM0, DSNCCOM1, or DSNCCOM2 to start or shut down the CICS attachment facility. To start the CICS attachment facility from an application program, link to DSN2COM0. To shut the CICS attachment facility down, link to DSN2COM2. Starting or stopping the CICS attachment facility by EXEC CICS START TRANSID('DSNC') is not supported.

If you have any applications that use the CICS EXTRACT EXIT command to determine whether the CICS attachment facility is active, you must change the command from:

```
EXEC CICS EXTRACT EXIT
 PROGRAM('DSNCEXT1') ENTRY('DSNCSQL')
```

to:

```
EXEC CICS EXTRACT EXIT
 PROGRAM('DSN2EXT1') ENTRY('DSNCSQL')
```

However, you might wish to replace the EXTRACT EXIT command with the INQUIRE EXITPROGRAM CONNECTST command. This has an advantage over EXTRACT EXIT in that it returns a value stating whether CICS is connected to DB2, irrespective of whether the exit is ENABLED or ENABLE STARTED. For more information on EXTRACT EXIT and INQUIRE EXITPROGRAM, see section 10.2.8, “Avoiding AEY9 Abends” on page 198.

2.4.8 Checklist

Here is a checklist for using the new CICS/ESA 4.1 attachment facility.

- Do not run the DSNTIJSU job when installing DB2 for the first time, or when you are upgrading your DB2 system.
- Do apply the preconditioning APARS to your DB2 system, if DB2 3.1 or DB2 2.3 is already installed.
- Do use the CICS installation JCL to install the new version of the CICS attachment facility.
- Do check for old attachment facility names in:
  - Your PLTPI and PLTSD
  - Your application programs
  - Any EXTRACT EXIT commands
  - Your CICS CSD file.
- Do reassemble your RCT.
  - Remember that RCT suffixes may now be two bytes or one byte
  - Don't put DB2 libraries in ASM.SYSLIB when you reassemble the table
  - Do linkedit the RCT with RMODE(24)
  - Decide whether to use new options
  - Update trace IDs if you have explicitly coded them in your existing RCT.

2.4.9 Installation Verification Procedure

The procedure is similar to that for the older CICS attachment facility shipped with DB2, and you should refer to 2.3.7, “Installation Verification Procedure” on page 34.

2.5 Summary of CICS Attachment Facility Modules

Figure 28 on page 46 shows a list of the old and new modules.

Old Attach Part Name	New Attach Part Name	Type
DSNCCMDP	DSN2CMDP	Module
DSNCCOM0	DSN2COM0	Module
DSNCCOM1	DSN2COM1	Module
DSNCCOM2	DSN2COM2	Module
DSNCEDF1	DSN2EDF1	Module
DSNCEDON	n/a	Module
DSNCEXT1	DSN2EXT1	Module
DSNCEXT2	DSN2EXT2	Module
DSNCEXT3	DSN2EXT3	Module
DSNCLI	DSNCLI	Module
DSNCMNFM	DSN2MNFM	Module
DSNCMSG0	DSN2MSG0	Module
DSNCMSGT	DSN2MSGT	Module
DSNCMSUB	DSN2MSUB	Module
DSNCPMSG	DSN2PMSG	Module
DSNCSAMG	DSN2SAMG	Module
DSNCSM21	n/a	Module
DSNCSM31	n/a	Module
DSNCSTOP	DSN2STOP	Module
DSNCSTRT	DSN2STRT	Module
DSNCUEXT	DSNCUEXT	Module
DSNCRCTx	DSN2CTxx	Load Module
DSNCRCT	DSNCRCT	Macro
DSNCPRMA	DSNCPRMA	Macro
DSNCPRMC	DSNCPRMC	Copybook
DSNCPRMP	DSNCPRMP	Copybook
DSNCSMPD	n/a	PL/AS copybook
DSNCUEPP	n/a	PL/AS copybook
DSN@UEXT	DSN@UEXT	Source

Figure 28. Summary of CICS Attachment Facility Modules

Chapter 3. Operation

In this chapter we discuss operating the CICS attachment facility and how to start and stop the CICS attachment facility. In Chapter 7, “Monitoring, Tuning, and Handling Deadlocks” on page 123 we discuss other important aspects of operating the CICS attachment facility. These include how to monitor the performance and status and how to enter DB2 commands.

To establish the operating environment:

- Appoint a person to be responsible for operating the CICS attachment facility. This would normally be the CICS master terminal operator.
- Appoint a person to be responsible for setting up operating procedures for the CICS attachment facility and related functions.
- Educate operators.

3.1 Establishing CICS Attachment Facility Operating Procedures

You need to develop some procedures to operate the CICS attachment facility. Include procedures to:

- Start the CICS attachment facility
- Stop the CICS attachment facility
- Handle messages
- Resolve in-doubt units of recovery (URs) after a failure of CICS, DB2, or the whole system
- Use CICS attachment facility commands to monitor the CICS attachment facility
- Enter DB2 commands.

You also need procedures for collecting data related to the CICS attachment facility for accounting, performance tuning, and problem determination. You could include procedures to:

- Start system monitoring facility (SMF) recording of DB2 output for use in accounting and tuning
- Use the CICS trace
- Start GTF recording DB2 output for use in accounting and tuning.

Note: Throughout this chapter, our examples use the complete syntax of CICS attachment facility and DB2 commands. 2.3.4.2, “Transaction Definitions” on page 27 shows how to use a shorter form by calling the function directly without prefixing such commands with DSNC.

3.2 Starting the CICS Attachment Facility

You establish the connection between CICS and DB2 by activating the CICS attachment facility. You can do this either automatically, at CICS initialization using the CICS PLTPI, or manually, using the DSNC STRT command from a CICS terminal.

3.2.1 Starting the CICS Attachment Facility Manually (before CICS/ESA 4.1)

You can use this command to start the connection to DB2. The full syntax of this command is DSNC STRT x The x in this command is the suffix for the RCT you want to use. The RCT loaded is DSNCRCTx. For example, if you enter the command DSNC STRT 1 RCT DSNCRCT1 is loaded. If you do not enter a suffix with the STRT command, the suffix defaults to 0 and RCT DSNCRCT0 is loaded.

The RCT defines the characteristics of the connection, including the identity of the DB2 subsystem that the CICS attachment facility connects to.

The RCT is loaded using the MVS program loader; so you do not have to define the RCT as a CICS program.

If the CICS attachment facility starts correctly, you see the message shown in Figure 29.

```
DSNC023I THE ATTACHMENT FACILITY HAS CONNECTED TO 'DSN' USING 'DSNCRCT0'
```

Figure 29. Successful Start of the CICS Attachment Facility (before CICS/ESA 4.1)

The message returned to the terminal issuing the start command identifies the DB2 system. In this case, it is DSN.

Figure 30 shows the message you see if the CICS attachment facility cannot find the RCT you specified on the DSNC STRT command. Usually you see this message because you either forgot to put a suffix on the command, or simply made a typing error. The message returned in this example tells us that the default RCT, DSNCRCT0, cannot be found.

```
DSNC002I   DSNCRCT '0' MODULE NOT FOUND
```

Figure 30. Error Starting the CICS Attachment Facility: Wrong RCT (before CICS/ESA 4.1)

3.2.2 Starting the CICS Attachment Facility Manually (CICS/ESA 4.1 Onward)

With the new CICS/ESA 4.1 attachment facility the full syntax of the DSNC STRT command is now: DSNC STRT rct-suffix,ssid.

With CICS/ESA 4.1 the default suffix for the DSNC STRT command is that specified in the INITPARM option (see 2.4.5.3, “Updating the System Initialization Table” on page 42 for a description of INITPARM, and Figure 24 on page 43 for an example of how to specify it). If you did not specify DB2 parameters in your INITPARM option, the RCT suffix default is 00 and the SSID defaults to that specified in the SUBID field of the RCT. Figure 31 on page 49 shows two examples of the DSNC STRT command with CICS/ESA 4.1. The first example starts the CICS attachment facility using RCT DSN2CT22. CICS is connected to the DB2 system named in the SUBID option of the TYPE=INIT macro of DSN2CT22. The second example starts the CICS attachment facility using RCT DSN2CT00. CICS is connected to the DB2 system named DB2A. You can see

that, as with the INITPARM option, the comma is used as a place holder to indicate that no RCT suffix is specified on the command

```
DSNC STRT 22

DSNC STRT ,DB2A
```

Figure 31. Examples of the DSNC STRT Command (CICS/ESA 4.1 Onward)

If the CICS attachment facility starts correctly, you see the message shown in Figure 32.

```
DSN2O23I THE ATTACHMENT FACILITY HAS CONNECTED TO 'DSN' USING 'DSN2CT22'
```

Figure 32. Successful Start of the CICS Attachment Facility (CICS/ESA 4.1 Onward)

The message returned to the terminal issuing the start command identifies the DB2 system. In this case, it is DSN.

Figure 33 shows the message you see if the CICS attachment facility cannot find the RCT. The message returned in this example tells us that the default RCT, DSN2CT00, cannot be found.

```
DSN2OO2I DSN2CT '00' MODULE NOT FOUND
```

Figure 33. Error Starting the CICS Attachment Facility: Wrong RCT (CICS/ESA 4.1 Onward)

Note that the messages in the new attachment facility are virtually the same as the messages in the old attachment facility, except that the new attachment facility messages start with DSN2. If you get a DSN2xxxx message, you can look it up under the DSNCxxxx message of the same number in the *IBM DATABASE 2 Messages and Codes* manual for your release of DB2. The few messages that are not documented in the *IBM DATABASE 2 Messages and Codes* under DSNC messages are listed in 3.4, "Handling Messages" on page 52.

3.2.3 Starting the CICS Attachment Facility Automatically

You can start the CICS attachment facility automatically, using the CICS PLTPI. To do this, you should add an entry for the command processor, DSNCCOM0, or DSN2COM0 to your PLTPI, if you already have one. (Refer to section 2.3.4.3, "Updating the PLTPI" on page 30 and section 2.4.5.1, "Updates to Your PLTPI and PLTSD" on page 41 for an explanation.) Make sure that the entry for your command processor comes after the entry for DFHDELIM (if you are using CICS/ESA 3.2.1 or later). If you do not already have a PLTPI, follow these steps:

1. Create a PLTPI (with suffix xx).
2. Include an entry for the CICS attachment facility command processor program, DSNCCOM0, or DSN2COM0, as appropriate.
3. Add a CICS program definition for this PLTPI, DFHPLTxx.
4. Check that DSNCCOM0 or DSN2COM0 is defined to CICS as a program.

5. Check that DSNC is defined to CICS as a transaction.
6. Add an entry to the SIT for the PLTPI, specifying the suffix of the PLT generated above (that is, PLTPI=xx).

Refer to section 2.3.4.3, “Updating the PLTPI” on page 30 and section 2.4.5.1, “Updates to Your PLTPI and PLTSD” on page 41 for further information on creating your PLTPI.

Note: You need a transaction definition for DSNC when using a PLTPI to activate the CICS attachment facility because this is the transaction ID attached when DSNCCOM0 or DSN2COM0 is invoked in this way. You may also use other transaction IDs to invoke DSNCCOM1 or DSN2COM1, to enter attachment and DB2 commands. See 2.3.4.2, “Transaction Definitions” on page 27.

3.2.4 Using Multiple Resource Control Tables

You may want to use multiple resource control tables (RCTs) if you have different CICS regions with different requirements from DB2 (for example, where the different CICS regions are used for production and testing, or where the work load running through the CICS systems is quite different). We recommend that you automatically start the CICS attachment facility in your production regions, using either the default RCT or specifying the RCT in the PARM parameter or in the INITPARM option. You may prefer to start the CICS attachment facility manually in your development and testing regions.

Note: A CICS region can be connected to only one DB2 subsystem at a time. If you want to connect to a different DB2 subsystem, you must stop the current connection using the DSNC STOP command, and then connect to the new DB2 subsystem using the DSNC STRT command. Alternatively, if you want to access data held in two DB2 subsystems, you can use the distributed database facilities of DB2 itself. In this case CICS is connected to one DB2 subsystem, and DB2 provides access to the data in the second DB2 subsystem.

If you enter the DSNC STRT command while your CICS attachment facility is active, the command is rejected and the message shown in Figure 34 is displayed on the terminal.

```
DSNC003I THE ATTACHMENT FACILITY IS ALREADY ACTIVE
```

Figure 34. Error Starting the CICS Attachment Facility: Already Active

Note: For CICS/ESA 4.1, message DSN2003I is displayed with the same text.

3.3 Stopping the CICS Attachment Facility

You can stop the the connection between CICS and DB2 either manually, using the DSNC STOP command, or automatically when the CICS region is shutdown, using the PLTSD. If you stop the connection manually, you may (under extreme circumstances) choose to force the CICS attachment facility to terminate.

3.3.1 Stopping the CICS Attachment Facility Manually: Normal Stop

You may stop the connection to DB2 at any time by using the command shown in Figure 35 from an authorized CICS terminal.

```
DSNC STOP QUIESCE
```

Figure 35. The DSNC STOP QUIESCE Command

The DSNC STOP QUIESCE command stops the CICS attachment facility after all currently running transactions are terminated. QUIESCE is the default option; DSNC STOP has the same effect. This command sets a flag in each entry in the CCT to signify that termination is in process. This ensures that:

- Any transaction that is waiting on a thread will be serviced before the subtask associated with that thread is detached.
- Any transactions entered after the STOP command will receive a nonzero return code when the first SQL statement is processed.

3.3.2 Stopping the CICS Attachment Facility Manually: Forced Stop

You may stop the connection by using the command shown in Figure 36 from an authorized CICS terminal.

```
DSNC STOP FORCE
```

Figure 36. The DSNC STOP FORCE Command

This form of the STOP command terminates the connection to DB2, waiting 15 seconds before detaching the thread subtasks. You should restrict use of the DSNC STOP FORCE command, because it can result in incomplete or in-doubt URs. These must be recovered by a subsequent restart of the connection.

3.3.3 Stopping the CICS Attachment Facility Automatically

The CICS attachment facility can also be terminated when CICS is terminated by using the CICS PLTSD. We recommend that you implement this in your CICS regions. If you already have a PLTSD, add an entry for DSNCCOM2 or DSN2COM2 (as appropriate) before the entry for DFHDELIM. If you do not already have a PLTPI, follow these steps:

1. Create a PLTSD (with suffix yy).
2. Include an entry for the CICS attachment facility program, DSNCCOM2 or DSN2COM2 as appropriate, before the DFHDELIM entry.
3. Add a CICS program definition for this PLTSD, DFHPLTyy.
4. Check that DSNCCOM2 (or DSN2COM2) is defined to CICS as a program.
5. Check that DSNC is defined to CICS as a transaction.
6. Add an entry to the SIT for the PLTSD, specifying the suffix of the PLT generated above (that is, PLTSD=yy).

As we described for the PLTPI, you must define DSNC as a CICS transaction, because DSNCCOM2 and DSN2COM2 attach this transaction ID when invoked from the PLTSD.

3.3.4 Termination Messages

Figure 37 shows the termination messages you receive after you issue a DSNC STOP command.

```
DSNC STOP

   DSNC012I   THE ATTACHMENT FACILITY STOP QUIESCE IS PROCEEDING


DSNC STOP FORCE

   DSNC022I   THE ATTACHMENT FACILITY STOP FORCE IS PROCEEDING
```

Figure 37. Examples of the CICS Attachment Facility STOP Commands and Messages

Figure 37 illustrates the two forms of the DSNC STOP command. The messages shown are sent to the terminal that issued the stop command. When termination is complete, one of the messages shown in Figure 38 is sent to the destination specified in SHDDEST option of the TYPE=INIT macro in the RCT for CICS attachment facility messages. (The default is CSMT.)

```
    DSNC025I THE ATTACHMENT FACILITY IS INACTIVE

    or

    DSN2025I THE ATTACHMENT FACILITY IS INACTIVE
```

Figure 38. CICS Attachment Facility Termination Complete

You can issue multiple DSNC STOP commands during termination processing. For example, a QUIESCE request can be followed by a FORCE request.

3.4 Handling Messages

New messages were added to both the older (DB2-supplied) and newer (CICS-supplied) versions of the CICS attachment facility.

3.4.1 Messages Added to the DB2-Supplied CICS Attachment Facility

Several new messages were added to the CICS attachment facility by APAR PN52110.

3.4.1.1 DSNC057I

Figure 39 shows the format of message DSNC057I.

```
DSNC057I csect-name CICS RELEASE release IS TOO HIGH FOR
THIS ATTACHMENT FACILITY
```

Figure 39. Message DSNC057I

Explanation: You are using a release of CICS that is too high for this CICS attachment facility, which requires CICS/ESA 3.3 or below. For this release of CICS, you must use the CICS attachment facility shipped with CICS.

System Action: The CICS attachment facility does not initialize.

User Response: Ensure that the CSD definition of the DSNC transaction specifies program DSN2COM1. If you are initializing the CICS attachment facility using a PLTPI, ensure that your PLTPI uses program DSN2COM0.

3.4.1.2 DSNC059I

Figure 40 shows the format of message DSNC059I.

```
DSNC059I csect-name CICS ATTACHMENT FACILITY MISMATCH WITH
RESOURCE CONTROL TABLE rct-name
```

Figure 40. Message DSNC059I

Explanation: The wrong macro library was used to assemble the RCT. This error message indicates that CICS is using the CICS attachment facility shipped with DB2, but the RCT was assembled using the CICS macro libraries.

System Action: The CICS attachment facility initialization is terminated.

User Response: Reassemble DSNCRCT using the DSNCRCT macro shipped in the DB2 macro libraries.

3.4.2 Messages Added to the CICS-Supplied Attachment Facility

The newer CICS attachment facility shipped with CICS/ESA 4.1 adds several new messages.

3.4.2.1 DSN2002I

Figure 41 shows the format of message DSN2002I.

```
DSN2002I csect-name DSN2CTxx MODULE NOT FOUND
```

Figure 41. Message DSN2002I

Explanation: The RCT DSN2CT with suffix xx could not be loaded.

System Action: The CICS attachment facility initialization is terminated.

User Response: Reenter the start command with the correct suffix. Ensure that the RCT DSN2CT is in the correct application program library, which is concatenated in the JOBLIB or STEPLIB statement of your CICS initialization JCL.

If DSN2CT is not in your application program library, ensure that you are using the correct version of the DSNCRCT macro to assemble DSN2CT. For CICS versions higher than CICS/ESA 3.3, the correct DSNCRCT macro is in the CICS macro library rather than in the DB2 macro library.

3.4.2.2 DSN2056I

Figure 42 shows the format of message DSN2056I.

```
DSN2056I csect-name CICS RELEASE release IS DOWN LEVEL FOR
THIS ATTACHMENT FACILITY
```

Figure 42. Message DSN2056I

Explanation: You are using a release of CICS that is down level for this CICS attachment facility, which requires CICS/ESA 4.1 or later. For this release of CICS, you must use the CICS attachment facility shipped with DB2.

System Action: The CICS attachment facility does not initialize.

User Response: Ensure that the CSD definition of the DSNC transaction specifies program DSNCCOM1. If you are initializing the CICS attachment facility using a CICS PLTPI, ensure that your PLTPI uses program DSNCCOM0 to initialize the attach.

3.4.2.3 DSN2059I

Figure 43 shows the format of message DSN2059I.

```
DSN2059I csect-name CICS ATTACHMENT FACILITY MISMATCH WITH
RESOURCE CONTROL TABLE rct-name
```

Figure 43. Message DSN2059I

Explanation: The wrong macro library was used to assemble DSN2CT, the RCT.

This error message indicates CICS is using the CICS attachment facility shipped with CICS, but the RCT was assembled using the DB2 macro libraries.

System Action: The CICS attachment facility is terminated.

User Response: Reassemble DSN2CT using the DSNCRCT macro shipped in the CICS macro libraries.

3.4.2.4 DSN2060I

Figure 44 shows the format of message DSN2060I.

```
DSN2060I csect-name INITPARM IS IGNORED BECAUSE FORMAT IS
INVALID.
```

Figure 44. Message DSN2060I

Explanation: The DSN2STRT INITPARM specified on the CICS initialization job is incorrect. The format of the INITPARM should be:

```
INITPARM=(DSN2STRT='xx,yyyy')
```

where xx is the suffix of the RCT and yyyy is the DB2 subsystem id. The RCT suffix cannot be greater than 2 characters, and the subsystem ID cannot be greater than 4 characters.

To specify only an RCT suffix, use the format:

```
INITPARM=(DSN2STRT='xx')
```

To specify only a subsystem ID override, use the format:

```
INITPARM=(DSN2STRT=',yyyy')
```

System Action: The CICS attachment facility attempts to initialize with the subsystem ID specified in the DSNC STRT command. If no subsystem ID is provided there, it attempts to initialize with the subsystem ID specified in the RCT.

User Response: If you want to override the subsystem ID, restart the CICS attachment facility using the DSNC STRT command with the correct SSID. Alternatively, you can correct the DSN2STRT INITPARM and reinitialize CICS.

3.4.2.5 DSN2061I

Figure 45 shows the format of message DSN2061I.

```
DSN2061I csect-name INITPARM IS INVALID.  ATTACHMENT
FACILITY NOT STARTED.
```

Figure 45. Message DSN2061I

Explanation: No RCT suffix was specified on the DSNC STRT command, and the format of the DSN2STRT INITPARM on the CICS initialization job is incorrect.

The format of the INITPARM should be:

```
INITPARM=(DSN2STRT='xx,yyyy')
```

where xx is the suffix of the RCT and yyyy is the DB2 subsystem id. The RCT suffix cannot be greater than 2 characters, and the subsystem ID cannot be greater than 4 characters. To specify only an RCT suffix, use the format:

```
INITPARM=(DSN2STRT='xx')
```

To specify only a subsystem ID override, use the format:

```
INITPARM=(DSN2STRT=',yyyy')
```

System Action: The CICS attachment facility does not initialize.

User Response: To start the CICS attachment facility, use the DSNC STRT command and specify an RCT suffix on the command. Alternatively, you can correct the DSN2STRT INITPARM and reinitialize CICS.

3.5 Manual Resolution of In-doubt Units of Recovery

In-doubt URs can occur when CICS, DB2, or the whole system fails while a transaction is carrying out syncpoint processing. In other words, a transaction started the syncpoint process using an EXEC CICS RETURN or EXEC CICS SYNCPOINT command, but that command had not completed at the time of the failure. In-doubt URs can also occur when the CICS attachment facility is terminated using a DSNC STOP FORCE command.

The in-doubt URs are normally resolved when the connection between CICS and DB2 is reestablished. CICS and DB2 exchange information about the in-doubt URs and resolve them. It is possible to lose the information needed to resolve the in-doubt URs. For example, if you perform a COLD start of CICS, the CICS logs are not used, and the information needed for an automatic recovery is lost. (CICS gets the information needed to resolve in-doubt URs from its logs during emergency restart.) If CICS and DB2 cannot resolve the in-doubt URs automatically, you need to resolve them manually. The process for this is described in detail in section 9.4.4, “Manual Resolution of In-doubt URs” on page 184.

3.6 Monitoring the Attachment Facility

You can monitor and modify the status of the connection between CICS and DB2 using commands provided by the CICS attachment facility. Below we summarize the commands that you can enter; they are described in detail in Chapter 7, “Monitoring, Tuning, and Handling Deadlocks” on page 123.

DSNC DISC (Disconnect) This command causes currently connected threads to be terminated as soon as they are not being used by a transaction.

DSNC DISP (Display) This command can be used to display:

- Information on the status of plan(s) defined in the RCT
- Information on the status of transaction(s) defined in the RCT
- Statistics associated with each entry in the RCT.

The output from the display commands is normally routed to the terminal that issued the command. Output can, however, be routed to other destinations.

DSNC MODI (Modify) This command allows characteristics of the connection to DB2 to be changed by modifying parameters in the RCT:

- Change the RCT-defined CICS destination where the attachment facility error messages have to be sent.
- For a given transaction ID, increase or decrease the value of THRDA, as defined in the RCT. The upper limit for this change is the THRDM value for this RCT group or entry.

3.7 Entering DB2 Commands

Once a connection between CICS and DB2 is established, terminal users authorized by CICS security can use the DSNC transaction to route commands to the DB2 system. These commands are in the form:

```
DSNC -DB2command
```

The command is routed to DB2 for processing. DB2 checks that the user is authorized to issue the command entered. Responses are routed back to the originating CICS user. The command recognition character (CRC) **-** must be used to distinguish DB2 commands from CICS attachment facility commands.

Both CICS and DB2 authorization are required to issue DB2 commands from a CICS terminal:

- CICS authorization is required to use the DSNC transaction, and
- DB2 authorization is required to issue DB2 commands.

Figure 46 on page 58 shows the CICS attachment facility commands (require CICS authorization to use the DSNC transaction), and the DB2 commands (require both CICS authorization to use the DSNC transaction, and DB2 authorization).

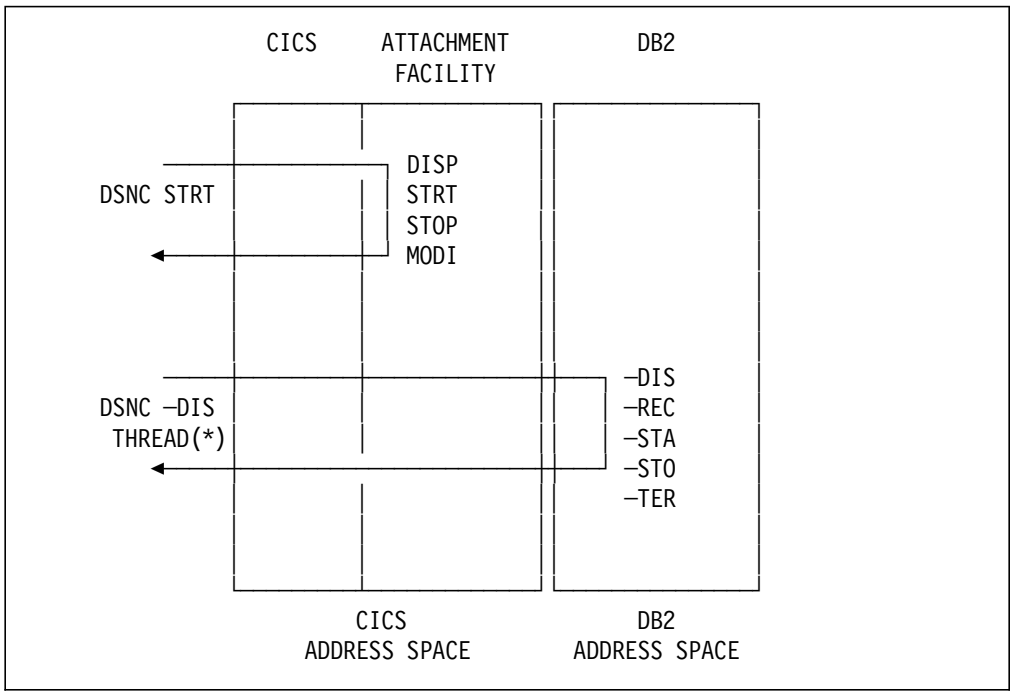


Figure 46. Examples of CICS Attachment Facility Commands and DB2 Commands

3.8 Starting System Monitoring Facility for Accounting, Statistics, and Performance

DB2, through its instrumentation facility, produces an SMF type 101 accounting record at each CICS DB2 thread termination and at each CICS DB2 sign-on, and an SMF type 100 record to hold DB2 statistics records on a DB2 subsystem basis. Performance and global trace records are produced as SMF type 102 records. These records can be directed to an SMF data set by:

- Specifying at DB2 installation that accounting or statistics data or both are required.

  Do this by setting the parameters MON, SMFACCT and/or SMFSTAT of the initialization macro DSN6SYSP to YES. Refer to the *IBM DATABASE 2 Administration Guide* for your release of DB2 for details.

- Activating SMF to write the records.

  Add entries for Type 101, 100, and 102 records to the existing SMF member (SMFPRMxx) entry in SYS1.PARMLIB.

- Starting the trace facility.

  Accounting, statistics, and performance traces can be started:

  – Automatically at DB2 start up time, by setting the parameters of the initialization macro DSN6SYSP to YES.

  - Dynamically by issuing the DB2 command -START TRACE, giving the specific trace and classes to be started. This is the way to start recording accounting, statistics, and performance data on a periodic basis.

- Activating SMF to write type 123 records using the SPQS or SPTS transactions.

DB2 does not produce any reports from the recorded data for accounting, monitoring, or performance purposes. To produce your own reports you can write your own programs to process this data, or use the DB2 Performance Monitor Program Product.

To use this facility for monitoring in a CICS-DB2 environment, refer to section 7.1.4.1, “Using the DB2 Statistics Facility” on page 135 and section 7.1.4.2, “Using the DB2 Accounting Facility” on page 136.

3.9 Starting and Stopping CICS Trace

The CICS auxiliary trace can be useful at times in solving programming or performance problems. If procedures are not already in place, you should establish procedures for starting, stopping, and listing the trace entries.

The CICS trace contains entries for each SQL statement, and for the PREPARE and COMMIT requests during syncpoint processing.

The *IBM DB2 Diagnosis Guide and Reference* describes how to interpret the different trace entries.

The *CICS/ESA Operations Guide* for CICS/ESA 3.3 and the *CICS/ESA Operations and Utilities Guide* for CICS/ESA 4.1 give details on how to use the CICS trace utility program.

3.10 Starting Generalized Trace Facility for Accounting, Statistics, and Performance

Instead of, or in addition to, recording accounting, statistics, and performance data to SMF, DB2 allows you to send this data to generalized trace facility (GTF). DB2 provides -START TRACE and -STOP TRACE commands. You can issue these through the CICS attachment facility DSNC transaction. You can use these commands to specify:

- Trace scope—Either GLOBAL for DB2 subsystem activity, PERFM for performance, ACCTG for accounting, STAT for statistics, or MONITOR for monitoring activity
- Trace destination—GTF, in main storage, or SMF, or both.
- Trace constraints—Specific plans, authorization IDs, classes, location, and resource managers.

Using the following procedure to use this trace with a GTF destination. Omit the GTF steps for a main storage trace.

1. Prepare:
   - Include a member in SYS1.PARMLIB, specifying GTF output for user records.

- Include a GTFDB2 start up procedure in SYS1.PROCLIB, referencing the above member.

2. Start the GTF trace by entering the MVS command START GTF.id.
3. Start the DB2 trace by entering the DB2 command -START TRACE=x DEST(GTF).
4. Perform monitoring activity.
5. Stop the DB2 trace by entering the DB2 command -STOP TRACE=x.
6. Stop the GTF trace by entering the MVS command P id with the same id used in the START GTF.id command.
7. List the GTF trace data set using the DB2 Performance Monitor Program Product facilities.

The *IBM DATABASE 2 Command and Utility Reference* gives detailed information on this trace.

Chapter 4. Security

You use several different security mechanisms to control access to resources in a CICS-DB2 environment:

- ***Dataset protection***—You can use the Resource Access Control Facility (RACF) to protect DB2 databases, logs, bootstrap data sets (BSDSs), and libraries outside the scope of DB2, from unauthorized access. You can also apply this protection to CICS data sets and libraries.

  You can use VSAM password protection as a partial replacement for the protection provided by RACF.

  There are no special considerations for using RACF to restrict access to libraries, databases, and data sets. We do not discuss this further.

- ***Connection authorization***—You can use RACF to verify that a subsystem (CICS, in this case) is authorized to connect to DB2.

- ***User authentication***—CICS sign-on, with or without RACF, authenticates users by checking that they supply a valid user ID and password.

  Note: DB2 does not authenticate the user; this must be done by CICS.

- ***Transaction authorization***—CICS checks that the authenticated user ID is authorized to use a specific transaction.

- ***DB2 security***—Restricts access to DB2 resources to authorized users only. DB2 resources include databases, tables, views, application plans, utilities, and commands.

Figure 47 on page 62 shows the security mechanisms involved in a CICS-DB2 environment.

4.1 Resource Access Control Facility Connection Authorization

If RACF is active, your CICS regions must be authorized to connect to DB2. If RACF is inactive, no connection verification is done.

DB2 invokes RACF when CICS tries to connect. It passes the CICS address space ID and the requested connection type. For CICS the connection type is single address space subsystem (SASS). RACF checks that the supplied ID is authorized to connect to DB2. The address space ID is the user ID of the JOB card, or the entry from the RACF table for started procedures (ICHRIN03). This is the initial primary authorization ID.

You must carry out the following steps to authorize CICS to connect to DB2:

1. Define a profile for the DB2 subsystem, with the type of connection allowed, in the RACF class DSNR.

   For example, the following RACF command creates a profile for SASS connections to DB2 subsytem DB2A in class DSNR.

```
RDEFINE DSNR (DB2A.SASS) OWNER(DB2OWNER)
```

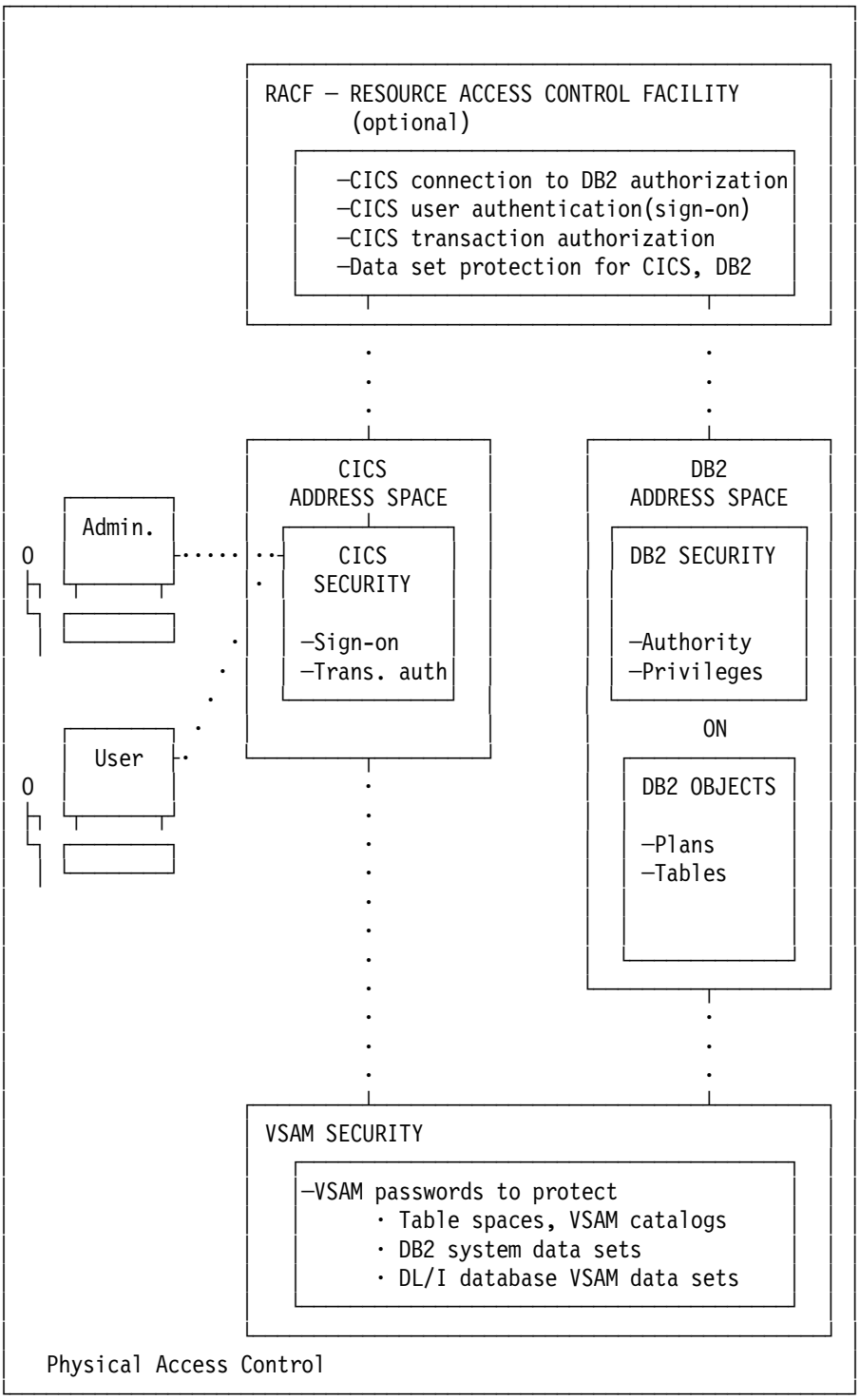


Figure 47. Overview of the CICS-DB2 Security Mechanisms

2. Permit CICS to access this DB2 subsystem.

   For example, the following RACF command permits CICS system CICSA11 to connect to DB2 subsystem DB2A:

```
PERMIT DB2A.SASS CLASS(DSNR) ID(CICSA11) ACCESS(READ)
```

If RACF verifies that this CICS region is allowed to connect to the DB2 subsystem, DB2 runs the connection authorization exit routine.

The initial primary authorization ID is always passed to the connection authorization exit routine. If you want to use DB2 secondary authorization IDs, you must replace the default connection authorization exit routine. If you want to use RACF group names as DB2 secondary authorization IDs, the easiest method is to use the IBM-supplied sample routine.

If SEC=YES is specified in the SIT and there is no RACF USER profile defined for the initial primary authorization ID, the CICS transactions abend. Note that even if the initial primary authorization ID is defined to RACF as a RESOURCE profile (for example as a terminal or transaction) the transaction still abends.

You must change the sample sign-on exit routine (DSN3SSGN) if all of the following conditions are true:

- You did not specify AUTH=GROUP in the RCT.
- The RACF list of groups processing is active.
- You have transactions whose initial primary authorization ID is not defined to RACF.
- The initial primary authorization ID was not defined to RACF as a user profile.

In summary, a CICS system using an RCT specification other than AUTH=USERID or AUTH=GROUP, cannot run the corresponding transactions when using the DB2 sample sign-on exit routine if the list of groups processing is active, unless the initial primary authorization ID is defined to RACF as a user.

To set up RACF for DB2 and for information about controlling access to a DB2 system, see *IBM DATABASE 2 Administration Guide*, Volume II.

4.2 User Authentication

If you are using CICS/ESA Version 3, you must either sign on to CICS to authenticate your user ID or permanently associate a user ID with a terminal using preset security. You need to authenticate your user ID if you want to restrict access to certain commands (for example DSNC) or to your own CICS-DB2 transactions. Additionally, you must sign on if you want to specify any of the following subparameters to the AUTH parameter in the RCT for authorization ID checking:

USERID
USER
GROUP

You can sign on using any method appropriate to your release of CICS, for example using the CESN transaction, or the EXEC CICS SIGNON command in an application program.

If you are using CICS/ESA Version 4, CICS passes the default user ID to DB2 if you do not sign on.

4.3 Transaction Authorization

CICS checks that your user ID is allowed to use a specific transaction. You should ensure that DSNC and any other transactions that you defined for specific CICS attachment facility tasks or DB2 commands are protected. See section 2.3.4.2, “Transaction Definitions” on page 27 for information on these transactions. You should also protect the transactions that invoke your own CICS-DB2 application programs.

4.3.1 CICS Attachment Facility Command Authorization

CICS attachment facility commands do not flow to DB2, and are subject only to the CICS transaction security mechanism. (CICS checks that you are authorized to use the DSNC transaction.) See Figure 46 on page 58. Thus, any user who is authorized by CICS transaction security to use the DSNC transaction can issue these commands.

DSNCCOM1 and DSN2COM1, which handle both CICS attachment facility and DB2 commands, can be activated by transactions other than DSNC (see section 2.3.4.2, “Transaction Definitions” on page 27). Thus you can give a different transaction code to each CICS attachment facility command:

DISC
DISP
STRT
STOP
MODI

4.3.2 DB2 Command Authorization

Unlike CICS attachment facility commands, which run entirely within CICS, DB2 command requests are routed to DB2. They are, therefore, subject to both the CICS transaction security mechanism (CICS checks that you are authorized to use the DSNC transaction) and to DB2 security checking. See Figure 46 on page 58. Any user who is authorized by CICS transaction security to use the DSNC transaction can issue these commands.

DSNCCOM1 and DSN2COM1, which handle both CICS attachment facility commands and DB2 commands, can be activated by transactions other than DSNC (see section 2.3.4.2, “Transaction Definitions” on page 27). Thus you can give a different transaction code to each DB2 command,

-DIS
-REC
-STA
-STO

Different CICS users can be given authorization to only a subset of these transaction codes. This means that individual users can be permitted to execute all, some, or none of the DB2 commands. CICS transaction security checking controls this.

Note: If you have access to the DSNC transaction, CICS allows you to issue all four of the DB2 commands.

You can distinguish DB2 commands from CICS attachment facility commands by the **-** character, which is entered with DB2 commands. This character is not a DB2 subsystem recognition character, but a command recognition character. It

is always a **-**, independent of the character actually defining the DB2 subsystem, since CICS can only connect to one DB2 subsystem at a time. There is no need to use different DB2 subsystem recognition characters from CICS, and thus we use only the default **-** character.

CICS checks that you are authorized to enter the DSNC transaction (or one of the other transactions that invokes DSNCCOM1 or DSN2COM1). DB2 security then checks that you are authorized to enter the particular DB2 command specified as data in the DSNC transaction (see section 4.4.2, "DB2 Command Authorization" on page 68).

4.4 DB2 Security

You have to be authorized to access resources within DB2. The authorizations you need differ for different tasks. You may need to be authorized for some, or all, of the following functions to carry out your assigned tasks:

- Enter DB2 commands from CICS terminals.
- Execute a plan using a CICS transaction.
- Execute dynamic SQL in a plan used by a CICS transaction.

Security checking in DB2 is based on your authorization ID.

4.4.1 DB2 Authorization IDs

Your authorization ID can be your CICS user ID, or it can be some other identifier (for example, your terminal ID, the transaction ID of the transaction you are using, or the CICS system ID). The authorization ID passed to DB2 is determined by the AUTH parameter in the RCT. For CICS, the valid authorization IDs are:

A character string
USERID (your CICS user ID)
USER (your CICS operator ID)
TERM (your terminal ID)
TXID (the transaction ID)
SIGNID (CICS system authorization ID)
GROUP (RACF group ID for signed-on user ID)

Specifying AUTH=GROUP is similar to the use of AUTH=USERID, but passes additional information to the DB2 sign-on authorization exit routine. AUTH=GROUP has the advantage that users can be allocated to, or deallocated from, group names as required. You can then grant DB2 authorizations to the group name, and you do not then need to change your DB2 authorizations when people join or leave groups. Authorization IDs are established at two different stages in CICS:

- The CICS recovery coordination task receives its authorization ID at connection time. Connection occurs when the CICS attachment facility is started.
- Transactions receive their authorization IDs at DB2 sign-on time. Section 1.3.4, "Transaction Sign-on" on page 11 describes how transactions sign on to DB2.

4.4.1.1 Primary and Secondary Authorization IDs

Every process that connects to DB2 is represented by one or more short identifiers called authorization IDs (in the case of CICS, the process is a CICS task). A process always has a primary authorization ID, and can optionally have one or more secondary authorization IDs. Privileges and authority can be granted to secondary authorization IDs. For example, a user can create a table using their secondary authorization ID. The table is then owned by that secondary authorization ID. Any other user with the same authorization ID has associated privileges over the table. To take privileges away from a user, simply disconnect the user from that authorization ID.

More generally, your CICS user ID can belong to a group of user IDs (defined to RACF). You may be able to perform actions in DB2 because your group user ID is authorized for the action, even if your individual user ID is not.

For CICS, secondary authorization IDs are implemented by the GROUP subparameter of the AUTH parameter in the RCT. Specifying GROUP is similar to specifying USERID, but additional information is passed to the DB2 sign-on authorization exit routine. The CICS attachment facility can:

- Interpret the GROUP subparameter in the RCT
- Pass the RACF user ID and the RACF group ID character strings to DB2 to be used for privilege and authority checking.

Note: You must replace the default CICS attachment facility authorization exit routines if you want to use secondary authorization IDs. The default authorization exit routines pass only a primary authorization ID to DB2. See section 4.4.1.2, “Authorization Exit Routines.”

4.4.1.2 Authorization Exit Routines

Primary and secondary authorization IDs can be either provided by RACF, or supplemented, changed, or both by an authorization exit routine. There are two authorization exits in DB2, one driven during connection processing and the other during sign-on processing.

DB2 provides default authorization exit routines for both exits; DSN3@ATH for the connection exit and DSN3@SGN for the sign-on exit. These default authorization exit modules are provided in object form only. The default authorization exits do not support secondary authorization IDs. You must replace them with your own routines, or with the sample authorization exits provided with DB2.

The DB2-supplied sample authorization exit routines are shipped in source form, to act as examples for your own routines. These samples are DSN3SATH and DSN3SSGN respectively. You can find them in DSN230.DSNSAMP or DSN310.SDSNSAMP.

If you specify the GROUP subparameter of the AUTH parameter in the RCT, the sample sign-on authorization exit routine passes a secondary authorization ID to DB2. If the RACF *list of groups* option is not active, the sample sign-on authorization exit routine passes the current connected group name as the only secondary authorization ID. If the RACF *list of groups* is active, the sample sign-on authorization exit routine obtains all the group names to which the user is connected and passes them to DB2 as secondary authorization IDs.

For more information about privileges and authorities, as related to controlling access to DB2 objects, refer to *IBM DATABASE 2 Administration Guide*. For more information about connection and sign-on exits and writing exit routines, see *IBM DATABASE 2 Administration Guide*. For more information about connection and the sign-on process as related to processing connections and processing sign-ons, see *IBM DATABASE 2 Administration Guide*.

4.4.1.3 Establishing Primary and Secondary Authorization IDs.

Remember, you must sign on to CICS (or establish your authorization ID in some other way, for example by using preset security for a terminal) if you specify any of the following subparameters to the AUTH parameter in the RCT for authorization ID checking:

USERID
USER
GROUP

Additionally, to use the GROUP option (and thus establish a secondary authorization ID):

- The CICS system must use RACF (or another external security manager). Specify SEC=YES in the SIT (or SEC=MIGRATE for CICS/ESA Version 3).
- If you are using the multiregion option (MRO), every connected CICS system should use RACF.
- Every CICS terminal user must sign on to CICS, for example by using the CESN transaction.
- If you are using MRO, sign-on information must be propagated from the TOR to the AORs. The CONNECTION definitions between the TOR and the AORs should specify ATTACHSEC=IDENTIFY to ensure that the user ID is propagated from one region to another.

If the conditions are met, the CICS attachment facility obtains the terminal user′s user ID and passes that to DB2 as the primary authorization ID. It also passes the RACF group name, or list of group names, as secondary authorization IDs.

If these conditions are not met, the CICS attachment facility issues an authorization failed message.

If GROUP was not specified, the secondary authorization ID is set to blanks. AUTH=(GROUP,USERID,Group_Name) is a valid specification.

Table 2 on page 68 shows the primary and secondary authorization IDs that the CICS attachment facility passes to DB2. CICS already checked that the terminal user is authorized to run the transaction by this stage.

Table 2. Effect of the RCT AUTH Parameter			
AUTH=	**Sign-on Required**	**Primary Authorization ID**	**Secondary Authorization ID**
GROUP	Yes	CICS user ID	RACF Groups
USERID	Yes	CICS user ID	BLANKS
USER	Yes	CICS operator ID	BLANKS
SIGNID	No	CICS system ID	BLANKS
TERM	No	CICS terminal ID	BLANKS
TXID	No	CICS transaction ID	BLANKS
character_ string	No	character_string	BLANKS

Note: `character_string` means that you can specify any character string for the AUTH parameter, for example, `AUTH=tester`. In this case, the CICS attachment facility passes the characters `tester` to DB2 as the primary authorization ID.

4.4.2 DB2 Command Authorization

In section 4.3.2, “DB2 Command Authorization” on page 64 we describe how CICS checks if you are authorized to issue a DB2 command. If CICS finds that you are authorized, your authorization IDs are passed to DB2 for further checking.

To authorize a user to enter a DB2 command you must GRANT DB2 command privileges to the authorization ID passed from CICS, because in contrast to the CICS attachment facility commands, DB2 checks the DB2 commands. The user must be authorized in CICS to execute the transaction code and in DB2 to execute the DB2 command.

In most cases, only a limited number of users are permitted to execute DB2 commands. A convenient solution can be to use the 8-byte CICS user ID as the authorization ID in DB2. You must specify this in the AUTH parameter in the TYPE=COMD macro in the RCT. Using this method, you can give different DB2 privileges explicitly to CICS user IDs. For example, you can use GRANT DISPLAY to explicitly give specific CICS user IDs permission to use only the -DIS command. Figure 48 on page 69 shows the security mechanisms for DB2 command authorization.

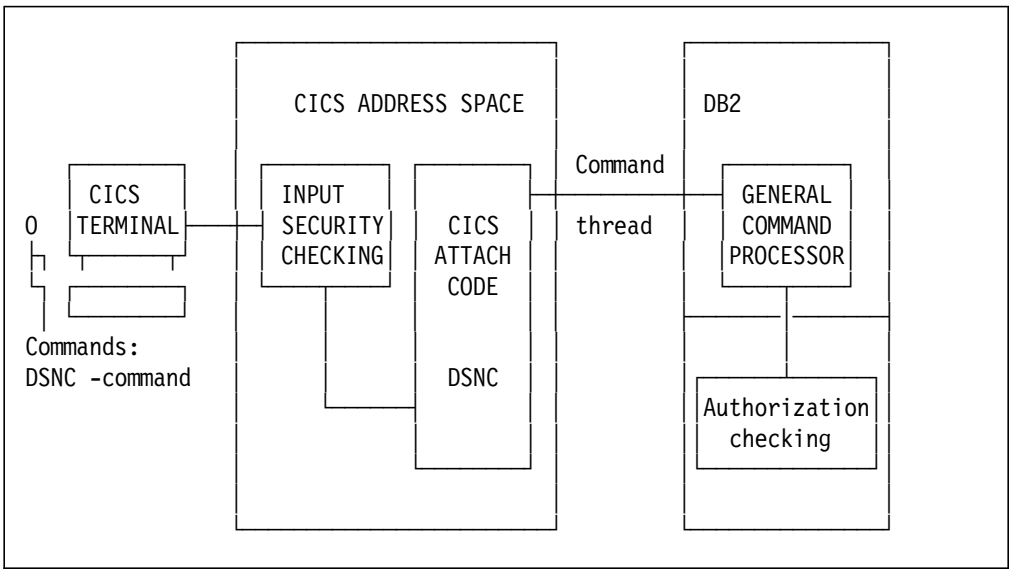


Figure 48. Security Mechanisms for DB2 Commands.

- *CICS security: Is the user authorized to execute the DSNC transaction?*
- *DB2 authorization: Is the user authorized to enter this DB2 command?*

In summary, authorization for the DSNC transaction is controlled through the use of:

- The AUTH= parameter on the DSNCRCT macro
- The EXTSEC and the TRANSEC parameter on the CICS transaction entry for DSNC
- The DB2 SYSOPR authority, which a user must have in order to use DB2 commands.

4.4.3 Plan Execution Authorization

Each CICS user entering a transaction that accesses DB2 needs CICS authorization to execute the transaction and DB2 authorization to execute the plan for the transaction. CICS transaction security checks that the user is authorized to use this transaction.

DB2 security relies on CICS authentication verification and does not provide an authentication mechanism. The authorization ID used to execute the DB2 plan for a transaction is defined in the TYPE=ENTRY or TYPE=POOL macro in the RCT. It can be any one of the authorization IDs listed in section 4.4.1, “DB2 Authorization IDs” on page 65. Figure 49 on page 70 shows how CICS checks that the user is authorized to run the transaction, and DB2 checks that the authorization ID is authorized to execute this plan.

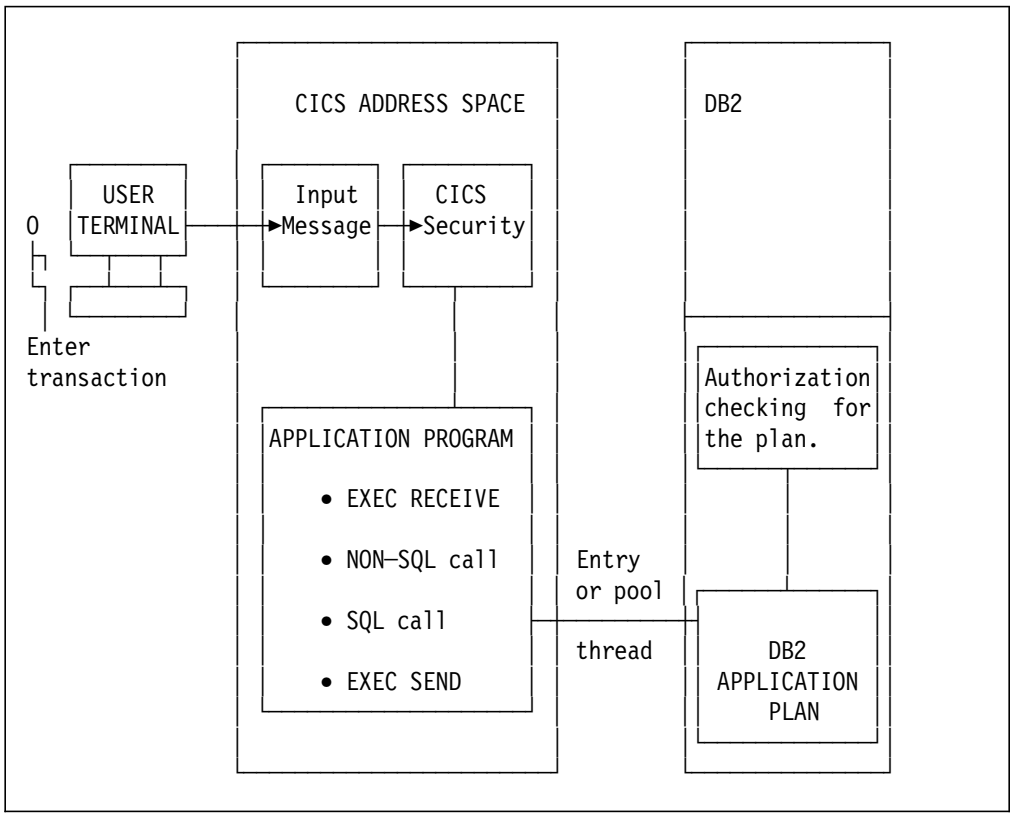


Figure 49. Security Mechanisms for Executing a Plan (Static SQL)

4.4.3.1 Performance Recommendation

If you have a large terminal network with many users, we recommend that you avoid using the following authorization IDs:

USERID
USER
TERM
GROUP

Using these authorization IDs can result in a number of disadvantages:

- Many GRANT statements are required.
- Many entries are in SYSPLANAUTH and SYSPACKAUT.
- Maintenance can be complicated.
- You can incur performance overhead.

If you use one of these authorization IDs, any CICS transaction using a DB2 thread is likely to have a different authorization ID from the last transaction that used the thread. This causes sign-on processing to occur. If you can arrange for the transactions using a thread to have the same authorization ID and transaction ID, sign-on authorization processing is bypassed. For best performance, use an AUTH of SIGNID, TXID, or a character string because these are constant, at least within a given transaction ID. You should use CICS security to restrict the use of the transaction, and hence execution of the plan, only to authorized users.

Note: Even if the authorization ID stays constant, certain SQL commands can cause a sign-on. Sign-on occurs when a transaction terminates if any of the DB2 special registers (CURRENT SQLID, CURRENT SERVER, and CURRENT PACKAGESET) do not match their initial value, or if there is an open cursor with hold. If you specify TOKENE=YES in the RCT, sign-on processing always takes place.

An alternative is to use GRANT to give EXECUTE authority on the plan to PUBLIC, because this also causes sign-on processing to be bypassed. This is not quite as efficient as using a constant authorization ID and transaction ID, because some processing still takes place in the CICS attachment facility. DB2 ignores the changed authorization ID.

4.4.3.2 Refreshing the CICS Attachment Facility Security Information

If the RACF access control environment element (ACEE) in the CICS region is changed in a way that affects the CICS attachment facility, DB2 is not aware of the change until the authorization ID changes and a sign-on occurs. This could happen if, for example, you connect a RACF user ID to a new GROUP (before APAR PN5557N). If you are using SIGNID, TXID, or a character string as your authorization ID, the only way to guarantee that DB2 becomes aware of the changes is to stop and then start the CICS attachment facility. Although this is a restriction, the performance costs of alternative designs would be too high.

There is no CICS attachment facility command equivalent to the CICS CEMT PERFORM SECURITY REBUILD to refresh security information.

4.4.3.3 Accounting

The authorization ID is used in each DB2 accounting record. The value of the authorization ID is the chosen identifier in the AUTH parameter. Depending on the TOKENE specification, it may not be possible to account at the individual user level.

From an accounting viewpoint the most detailed information is obtained if using USERID, USER, GROUP or TERM. These options are, however, not the recommended options from a performance viewpoint, as explained in the previous section.

For more information about accounting in a CICS-DB2 environment, see Chapter 8, “Accounting” on page 153.

4.4.4 Static SQL

When using static SQL, the binder of the plan must have the privileges needed to access the data, and the authorization ID passed from CICS to DB2 need only have the privileges to execute the plan.

Synonyms can be useful when moving from a test to a production environment. You can use them to identify tables and views when using unqualified SQL. If the SQL statements in your programs are not qualified, DB2 uses the authorization ID of the plan′s binder as the qualifying part of the statement at bind time unless you created a synonym for the binder; in that case, the synonym is used. The synonym then specifies the fully qualified table name or view name.

Notes:

1. You cannot create a synonym on behalf of another user.
2. The synonym is resolved at bind time.
3. If a synonym is dropped, all plans based at the synonym are invalidated.
4. If a plan is bound with VALIDATE(RUN) and the synonym did not exist at bind time, the bind process is completed at execution time. The original binder is still the binder when the bind process is completed at execution time. It is not the authorization ID specified in the AUTH parameter of the RCT. The synonym must be created under the authorization ID of the original binder.
5. During the bind process at execution time DB2 will:
   - Check authorization
   - Select path
   - Generate execution code

   and other necessary functions to build the plan. This involves overhead at execution time.

We recommend that you do not use both synonyms and VALIDATE(RUN) together.

4.4.5 Dynamic SQL

When using dynamic SQL, the authorization ID passed from CICS to DB2 must possess the privileges required to access the DB2 resources involved. If, for example, we specify AUTH=USERID in the RCT, the CICS user ID must be granted DB2 privileges to the DB2 resources involved in the dynamic SQL. If this user ID is also a TSO-id, it has access to the DB2 resources directly from SPUFI, QMF, and other utilities.

We recommend you use either the SIGNID or a character string as the authorization ID for dynamic SQL. This limits the number of GRANT statements you need to make. In most cases, you will find it convenient to have just one authorization ID under which all dynamic SQL in CICS is executed.

4.4.6 Qualified or Unqualified SQL

When you write programs you have to decide whether to use qualified SQL or unqualified SQL. We recommend you use qualified SQL for dynamic SQL statements, because it is easier to administer.

4.4.6.1 Using Unqualified SQL

If you use unqualified SQL, you must decide how to supply the CREATOR to fully identify the tables and views. There are two possibilities:

- You can use synonyms. The synonym must be created by the authorization ID specified in the RCT. Synonyms can only be created by the authorization ID itself. That means that you must develop a method to create the synonyms. You can use a TSO-id with the same ID as the authorization ID specified in the RCT. Another possibility is to design a CICS transaction (using the same authorization ID) that itself could do the CREATE SYNONYM statement. However, neither of these methods are desirable.
- If you do not use synonyms, the CREATOR used in the bind process is the authorization ID of the binder. All tables and views referenced in the

dynamic SQL must then be created with this ID. All transactions using dynamic SQL to access a common set of DB2 resources must then have the same authorization ID specified in the RCT. In most cases, it must be the SIGNID, or a character string. This restriction is normally not acceptable.

For these reasons, we do not recommend the use of unqualified SQL in dynamic SQL statements.

Descriptions of DB2 security and authorization can be found in the International Technical Support Organization, San Jose Center, publication *DB2 Version 2.1 Security and Authorization Extensions Guide*.

Chapter 5. Defining the Resource Control Table

The connection between CICS and DB2 is defined in the resource control table (RCT). This table describes the relationship between CICS transactions and DB2 resources (application plans and command processors). It also defines the connection between CICS and DB2.

The RCT is generated by a macro (DSNCRCT) supplied by the CICS attachment facility. For a detailed description of this macro and how to use it when connecting the CICS attachment facility, see the *IBM DATABASE 2 Administration Guide*.

In this chapter we give a general description of the RCT, and the DSNRCT macro, including:

- A description of the different connection types that exist between CICS and DB2
- A summary of the main activities involved in processing SQL requests for typical connection types
- A summary of the parameters defining the RCT
- Recommendations for specifying the RCT
- A description of how different thread types are created, used, and terminated. This section includes recommended RCT and BIND options for typical transaction types
- A series of examples on how to define the RCT in a specific environment.

5.1 CICS-DB2 Connection Types

There are three main types of threads that potentially can be used by the CICS attachment facility:

Command threads
Pool threads
Entry threads.

These thread types only relate to the kind of thread that is actually used by a DB2 command or a transaction. The definition of threads in the RCT, using the TYPE=COMD, TYPE=POOL, and TYPE=ENTRY macro specifications, do not have a one-to-one relationship with the three main type threads.

The three main thread types are defined below:

Command threads DB2 command threads are reserved for command processing through the DSNC transaction. They are used for processing DB2 commands only. They are not used for the CICS attachment facility commands, because these commands are not passed to DB2.

Requests for a DB2 command thread can be transferred to the pool if a DB2 command thread is not available.

Pool threads Pool threads are used for all transactions and commands not using an entry thread or a DB2 command thread. Pool threads are normally used for low volume transactions and

overflow transactions from entry threads and DB2 command threads. A pool thread is terminated immediately when it is unused.

Entry threads These threads are intended for high-volume transactions, high-priority transactions, and controlled transactions. Entry threads can be defined as protected or unprotected. See section 5.3, “Coordinating RCT and BIND Options” on page 80 for the definition of a controlled transaction.

Protected entry threads are not terminated for a period even if they are unused. Many CICS transactions can use the same protected entry thread and avoid the overhead involved in creating and terminating the thread for each transaction.

Unprotected entry threads terminate if no CICS transaction uses them. The overhead of creating and terminating the thread must be taken into account for transactions using unprotected entry threads.

The MVS subtask for an entry thread is not terminated, even if the entry thread is terminated. This is true for both protected and unprotected entry threads. The MVS subtask is only terminated if the number of TCBs is equal to THRDMAX-2, or if the CICS attachment facility terminates.

Requests for an entry thread can be transferred to the pool, if an entry thread is not available.

5.2 Main Activities in SQL Processing

You should select the thread type to use for a given set of transactions together with the BIND parameters for the corresponding plan. This is because the BIND parameters determine whether a number of activities are related to the thread or to the transactions.

The main DB2 activities involved in processing CICS transactions with SQL calls are:

- Create a thread or reuse an existing thread.
- Process the SQL statements.
- Perform commit processing.
- Terminate the thread or keep it.

These main activities are performed for each transaction. The work involved in each activity depends on a number of options, which you define. The most important options are:

- RCT specification
  - Use of protected entry threads
  - Use of unprotected entry or pool threads
  - Authorization strategy.

- BIND options
  - ACQUIRE
  - RELEASE
  - VALIDATE

- Use of PACKAGES

5.2.1 Thread Creation

The following activities can occur at thread creation time, depending on the BIND options (sign-on and authorization check are always performed at thread creation time):

- Sign-on
- Check the maximum number of threads.
- For an application plan, load the skeleton cursor table (SKCT) and header, if not already in the environmental description manager (EDM) pool.
- Create a copy of the SKCT header in the cursor table (CT).
- Perform the plan authorization check.
- If ACQUIRE(ALLOCATE) is specified:
  - Acquire table space (TS) locks for all TSs referenced in the plan.
  - Load all data base descriptors (DBDs) referenced in the plan.

If you use protected threads, you eliminate the work required for thread creation and termination for individual transactions. A protected thread is not terminated when a transaction ends, and the next transaction associated with the same TYPE=ENTRY macro in the RCT reuses the thread. We recommend that transactions using protected threads should use a plan bound with ACQUIRE(ALLOCATE) and RELEASE(DEALLOCATE) to further reduce the amount of work done.

Note: Although ACQUIRE(ALLOCATE) and RELEASE(DEALLOCATE) reduce the amount of processing for a protected thread, there are some locking considerations. We discuss these further in section 5.3.3, “Locking” on page 81.

Packages do not have a separate ACQUIRE option. ACQUIRE(USE) is implicit when a program is executed from a package.

Infrequently-used transactions should not use a protected thread, because the thread terminates between two such transactions. The plans for these transactions should not typically use the BIND parameter ACQUIRE(ALLOCATE), unless most of the SQL statements in the plan are used in each transaction.

The control block for an application plan, the SKCT, is divided into sections. The header and directory of an SKCT contain control information; SQL sections contain SQL statements from the application. A copy of the SKCT, the CT, is made for each thread executing the plan. Only the header and directory are loaded when the thread is created, if they are not already in the EDM pool.

Note that the SQL sections of the plan segments are never copied into the CT at thread creation time. They are copied in, section by section, when the corresponding SQL statements are executed. For a protected thread with RELEASE(DEALLOCATE), the CT increases in size until all segments have been copied into the CT.

If the SQL statements for the transaction are bound to a package rather than a plan, DB2 uses a skeleton package table (SKPT) rather than a SKCT and a

package table (PT) rather than a CT. The SKPT is allocated when the first SQL statement is executed; it is not allocated when the thread is created.

5.2.2 SQL Processing

The following activities can occur for each SQL statement processed, depending on thread reuse and the BIND options.

- For the first SQL call in a transaction reusing a thread with a new authorization ID:
  - Sign-on
  - Authorization check.
- Load the SKCT SQL section, if it is not already in the EDM pool.
- Create a copy of the SKCT SQL section in the CT, if it is not already there.
- If ACQUIRE(USE) is specified:
  - Acquire referenced TS locks, if not already taken.
  - Load DBDs in EDM pool, if they are not already there.
- Process the SQL statement.

You can minimize the work done at SQL processing time by using ACQUIRE(ALLOCATE).

If the SQL statement is in a package, the SKPT directory and header are loaded. The PT is allocated at statement execution time, rather than at thread creation time, as in the case with the SKCT and the CT for plans bound using ACQUIRE(ALLOCATE). The authorization ID for the package is checked in SYSPACKAUTH, using index DSNAPH01. Authorization checking also occurs when the SQL statement is executed.

The authorization ID for an application plan is checked in the SYSPLANAUTH catalog table. If this check fails, the table SYSUSERAUTH is checked for the SYSADM special privilege.

5.2.3 Commit Processing

The following activities can occur at commit time, depending on your BIND options:

- Release the page locks.
- If RELEASE(COMMIT) is specified:
  - Release the TS locks
  - Free the CT pages.

You can minimize the work done at commit time by using RELEASE(DEALLOCATE). Using RELEASE(DEALLOCATE) optimizes performance if there are many commits in the program and if the thread is to be reused, because the work associated with releasing the TS locks and freeing the CT pages is done once, when the thread terminates, and not for each COMMIT statement.

Transactions release the thread they are using at different times, depending on whether they are running at a terminal or not. If the transaction is terminal oriented, the thread is released at SYNCPOINT or at end of task (EOT). This makes it efficient to use a protected thread for terminal-oriented transactions

issuing many COMMITS, if combined with the BIND options ACQUIRE(ALLOCATE) and RELEASE(DEALLOCATE). In this case the resources to do the following activities are saved for each commit:

- Terminate and start the thread
- Release and acquire the TS locks
- Release and copy segments of the plan into the CT.

Terminal-oriented threads are not released at SYNCPOINT time if:

- Held cursors are open
- Modifiable special registers are not at their initial value. Table 3 shows the DB2 modifiable special registers. Table 4 shows the DB2 unmodifiable special registers.

If the transaction is not terminal-oriented, the thread is released at EOT only. In this case you do not need to use a protected thread to get thread reuse after an EXEC CICS SYNCPOINT command. You may still want to use a protected thread if this is a frequently used transaction. The BIND options ACQUIRE(ALLOCATE) and RELEASE(DEALLOCATE) give the same advantages as for the terminal-oriented transactions with many commits.

Table 3. DB2 Modifiable Special Registers

Modifiable Special Registers	**Introduced in:**	**Comments**
CURRENT PACKAGESET	DB2 Version 2	
CURRENT SQLID	DB2 Version 2	
CURRENT DEGREE	DB2 Version 3	
CURRENT SERVER	DB2 Version 3	CONNECT TYPE-2

Table 4. DB2 Unmodifiable Special Registers

Unmodifiable Special Registers	**Introduced in:**	**Comments**
CURRENT DATE	DB2 Version 2	
CURRENT TIME	DB2 Version 2	
CURRENT TIMESTAMP	DB2 Version 3	
CURRENT TIMEZONE	DB2 Version 3	
CURRENT USER	DB2 Version 2	

5.2.4 Thread Termination

The following activities can occur at thread termination time, depending on the BIND options:

- If RELEASE(DEALLOCATE) is specified:
  - Release TS locks
  - Free CT pages
- Free work storage.

5.3 Coordinating RCT and BIND Options

You can create many different combinations of RCT and the BIND options. One of the most important things you need to do to optimize performance is to define whether a given set of transactions should use one or more protected threads. You should then define the BIND parameters ACQUIRE and RELEASE to minimize the total amount of work related to the four main activities involved in processing SQL transactions. It is important that you coordinate your RCT and BIND options.

5.3.1 Recommended Combinations

In general we recommend that you initially set these options to the values shown in Table 5. You may find that you will get better performance from other combinations for your own transactions. For each transaction type we show a recommended thread type and BIND option. We also recommend whether transactions should overflow to the pool.

Table 5. CICS-DB2 Transaction Types. The table shows some typical transaction types and the recommended RCT and BIND options.

Transaction Description	Thread Type	Overflow	ACQUIRE	RELEASE
High volume (all types)	Protected Entry	Note 1	ALLOCATE	DEALLOCATE
Terminal-oriented with many commits	Protected Entry	Note 2	Note 3	DEALLOCATE
Low volume, high priority	Unprotected Entry	Yes	USE	COMMIT
Low volume, limited concurrency	Unprotected Entry	Never	USE	COMMIT
Low volume, low priority	Pool	Not Applicable	USE	COMMIT
Nonterminal oriented with many commits	Note 4	Note 4	Note 3	DEALLOCATE

Notes:

1. Yes, but define enough entry threads so this happens infrequently
2. Yes, but if it overflows to the pool no protected thread is used.
3. ALLOCATE if most of the SQL in the plan is used. Otherwise use ACQUIRE(USE).
4. Threads are held until EOT. Use pool threads for a short transaction. Consider entry threads for longer running transactions.

In Table 5 limited concurrency means only a limited number (n) of transactions are allowed to execute at the same time. A special case exists when n=1. In that case the transactions are serialized. You can still use a protected thread if the transaction rate is high enough to make it worthwhile. The transactions cannot be controlled, if overflow to the pool is allowed. You should normally use the CICS mechanism for limiting the number of transactions running in a specific class, rather than forcing transactions to queue for a limited number of threads.

As Table 5 shows, only a few combinations of RCT and BIND options are generally recommended. However, in specific situations other combinations can be used.

5.3.2 Processing SQL requests

Table 6 shows a summary of the activities involved in processing SQL requests for the three recommended sets of RCT and BIND specifications.

Table 6. Thread-Related Activities. The table shows the main activities involved in processing DB2 transactions for recommended combinations of RCT and BIND options.

Activity	Protected Threads		Unprotected Threads	
	ACQUIRE(ALLOCATE) RELEASE(DEALLOCATE)		(USE) (COMMIT)	(USE) (DEALLOCATE)
	Activity for each thread	Activity for each transaction	Activity for each transaction	Activity for each transaction
Create thread:	X		X	X
SIGNON	X	(1)	X	X
Authorization Check	X	(1)	X	X
Load SKCT Header	X		X	X
Load CT Header	X		X	X
Acquire all TS locks	X			
Load all DBDs	X			
For each SQL statement:				
Load SKCT SQL section		(2)	(2)	(2)
Create CT copy		(3)	X	X
Acquire all TS locks			X	X
Load all DBDs			X	X
Commit:				
Release page locks		X	X	X
Release TS locks			X	
Free CT pages			X	
Terminate Thread:	X		X	X
Release TS locks	X		X	X
Free CT pages	X			X
Free work storage	X		X	X

Notes:

X. Required activity
1. Only if new authorization ID ID
2. Only if SQL section is not already in EDM pool
3. Only if SQL section is not already in Cursor Table

Table 6 illustrates the performance advantage of using protected threads without changing the authorization ID.

5.3.3 Locking

DB2 uses a lock mechanism to control concurrency while maintaining data integrity. To maximize concurrency in the CICS environment, you should use page locking instead of table space locking. You can obtain this by defining: LOCKSIZE(PAGE) or LOCKSIZE(ANY) when creating the table space and the isolation level as cursor stability at BIND time.

The specification of LOCKSIZE(ANY) allows DB2 to decide if lock escalation can take place for the table space. If the number of locks exceeds NUMLKTS, lock escalation takes place. The DB2 parameter NUMLKTS is the number of concurrent locks per table space. NUMLKTS should then be set to a value so

high that lock escalation does not take place for normal CICS operation. If a table space lock is achieved and the plan was bound with RELEASE(DEALLOCATE), the table space will not be released at COMMIT time as only page locks are released. This can mean that a thread and plan monopolizes use of the table space.

A reason for using ANY instead of PAGE is to give DB2 the option to use lock escalation for programs that acquire many page locks before committing. This is typically the case with batch programs.

You can override DB2 rules for choosing initial lock attributes by using the SQL statement LOCK TABLE in an application program.

You should avoid the LOCK TABLE statement, unless it is strictly necessary. If the LOCK TABLE statement is used, the plan should be bound RELEASE(COMMIT). In fact, if the LOCK TABLE is used in an online program, it can exclude the use of RELEASE(DEALLOCATE) and protected threads.

5.4 Resource Control Table Parameters

This section summarizes the parameters that can be specified in the RCT either for the old CICS attachment facility provided with DB2 2.3 and DB2 3.1, or the new CICS attachment facility provided with CICS/ESA Version 4. We make recommendations on which parameters to use and what to specify for them in section 5.5, “Recommendations for RCT Specifications” on page 84. The parameters are grouped TYPE= statements in the RCT. These are:

TYPE=INIT
TYPE=COMD
TYPE=ENTRY and TYPE=POOL
TYPE=FINAL

5.4.1 TYPE=INIT Statement

Two kinds of information can be specified in the TYPE=INIT statement. One set of parameters is used to define the connection between CICS and DB2. Another set of parameters is used to specify default values for the TYPE=COMD, TYPE=POOL, and TYPE=ENTRY statements.

5.4.1.1 Parameters for Defining the CICS-DB2 Connection

ERRDEST Specifies up to three CICS transient data destinations to receive unsolicited messages.

PCTEROP Specifies the type of processing that is to occur following a create thread error.

PLNPGMI Specifies the name of the default dynamic plan exit.

PLNXTR1 Specifies the dynamic plan *entry* trace id.

PLNXTR2 Specifies the dynamic plan *exit* trace id.

PURGEC (for CICS/ESA Version 4 only)
Sets the protected thread purge cycle.

SNAP Specifies the MVS SYSOUT class to use for taking a SNAP dump if a thread subtask fails. You cannot specify a data set name, only the SYSOUT class.

STRTWT Specifies the option to take during the CICS attachment facility startup process if DB2 is not active.

SUBID Specifies the name of the DB2 subsystem where CICS is to be connected.

SHDDEST Specifies the transient data destination to receive the statistical report produced when the CICS attachment facility is being stopped.

SIGNID Specifies the authorization ID that can be used when a thread task signs on.

SUFFIX Specifies the suffix characters that identify this RCT.

THRDMAX
Specifies the maximum number of threads that could be created between CICS and DB2 within this RCT.

TRACEID Specifies the CICS user trace identification that the CICS attachment facility is to use for tracing calls to DB2.

TXIDSO (for CICS/ESA Version 4 only)
Allows suppression of some sign-ons during thread reuse to prevent extraneous accounting records from being written.

5.4.1.2 Parameters for Thread Definition Defaults

DPMODI Specifies the default priority for the thread task compared to the CICS main task.

PLANI (for CICS/ESA Version 4 only)
Specifies the default plan.

ROLBI Specifies the default action for deadlock and timeout situations.

TOKENI Specifies the default action for producing account records for every transaction.

TWAITI Specifies the default action for when a thread is not available.

5.4.2 TYPE=COMD Statement

This type of thread definition is needed only if the default values for the command thread have to be changed. If needed, specify it immediately after the TYPE=INIT statement and before the TYPE=POOL statement.

AUTH Specifies the authorization option.

ROLBE Specifies the action for deadlock and timeout situations. We recommend ROLBE=NO for the TYPE=COMD form because rollback is not used for DB2 commands.

DPMODE Specifies the priority for the thread task in relation to the CICS main task.

TASKREQ Specifies the transaction identification for this entry when the transaction is started via one of the 3270 function or attention keys.

THRDM Defines the maximum value that you can specify for THRDA.

THRDA Specifies the maximum number of active DB2 command threads.

THRDS Specifies the number of MVS subtasks to be attached for the command threads when the attachment facility is started. It also specifies the number of protected command threads.

TXID Specifies the transactions that should use this command thread.

TWAIT Specifies the action if no thread is available for an SQL request.

5.4.3 TYPE=POOL and TYPE=ENTRY Statement

The parameters for pool and entry definitions are almost identical. They are listed here:

AUTH Specifies the authorization option.

DPMODE Specifies the priority of the thread TCB compared to the CICS main TCB.

PLAN Specifies the name of the plan to use.

PLNEXIT Specifies whether to invoke a dynamic plan allocation exit.

PLNPGME
Specifies the name of the exit program this entry uses.

ROLBE Specifies the action for deadlocks and timeouts.

TASKREQ Specifies the transaction identification for this entry when the transaction is started via one of the 3270 function or attention keys. *This parameter is for entry threads only.*

THRDM Defines the maximum value to which THRDA can be set.

THRDA Specifies the current maximum number of active threads for this RCT entry. You can change THRDA dynamically using the DSNC MODIFY command.

THRDS Specifies the number of MVS subtasks to be attached for this RCT entry when the CICS attachment facility is started. It also specifies the number of protected threads.

TOKENE Specifies the default action for producing account records for every transaction. *This parameter is for entry threads only.*

TWAIT Specifies the action for when no thread is available for an SQL request.

TXID Specifies the transaction(s) using this entry.

5.4.4 TYPE = FINAL Statement

You do not specify any parameters for the TYPE=FINAL statement.

5.5 Recommendations for RCT Specifications

Refer to the *IBM DATABASE 2 Administration Guide* for a detailed description of all RCT parameters.

5.5.1 TYPE=INIT Statement

This section deals with the RCT parameters SUFFIX, THRDMAX, PURGEC and PLANI.

5.5.1.1 SUFFIX

If more than one CICS is connected to the same DB2 system, they are likely to use different RCTs.

For the CICS attachment facility provided with DB2 2.3 and DB2 3.1, the RCT suffix is a single byte. The default value is SUFFIX=0. To specify the RCT to be used, you enter the suffix as part of the DSNC STRT command (DSNC STRT x, where x is the suffix of the RCT you want to use), or you specify the suffix when starting the CICS attachment facility using a PLTPI. If you are using the PLTPI, specify the character string DSNCRCTx as the first parameter on the PARM field of the CICS EXEC DFHSIP JCL, where x represents the suffix of the desired RCT. For example:

```
//CICS EXEC PGM=DFHSIP,PARM=('DSNCRCT2,SIT=33,SYSIN')
```

For the CICS attachment facility provided with CICS/ESA Version 4, the RCT suffix is two bytes, and the default is now SUFIX=00. You can issue the DSNC STRT command from a terminal logged on to CICS, as before. However, if you use the PLTPI process, the old DSNCRCTx parameter on the CICS initialization JCL is now the CICS INITPARM parameter. You can specify the INITPARM either on the PARM= parameter of the EXEC PGM=DFHSIP statement of your initialization JCL, or in the SYSIN dataset. The DSNC STRT command has a new format with CICS/ESA Version 4, where you can specify both the rct-suffix and the DB2 subsystem that CICS is to connect to, thus overriding the SUBID parameter of the TYPE=INIT statement. Refer to section 2.4.5, “Updating CICS Definitions and Tables” on page 41 and section 3.2, “Starting the CICS Attachment Facility” on page 47 for details.

5.5.1.2 THRDMAX

For performance reasons, you should check that the sum of THRDA from all COMD, ENTRY, and POOL threads is *less* than or *equal to* THRDMAX-2. Be aware that when you do not explicitly specify the COMD thread, it has a THRDA value of one.

The maximum number is provided to control the number of MVS subtasks for each active thread. When the number of THRDMAX-2 is reached, and there is a demand for more subtasks, the CICS attachment facility terminates all the subtasks that are currently inactive. For that reason, the recommended value for THRDMAX is the sum of all values on the THRDA parameters + 3.

You can dynamically increase the THRDA values using the DSNC MODIFY command, up to the value of THRDM. You should take this into account when specifying the THRDMAX value.

The CTHREAD parameter of the DB2 DSNZPARM controls the number of concurrently active threads DB2 allows. The sum of all active threads from TSO users, all CICS and IMS/VS systems, and other systems accessing DB2 should not exceed CTHREAD. Otherwise, the result can be unpredictable response times. If the CTHREAD value is reached, a CICS attachment facility request to create a thread is queued by DB2, and the CICS transaction is placed in wait state until a thread is available. If you increase the THRDMAX value or set up an additional CICS systems with access to the same DB2 system, you may need to increase the CTHREAD parameter. This should be carefully monitored.

You should try to minimize the total number of DB2 threads in your CICS system. Unnecessary threads (TCBs) cause extra MVS dispatching overhead and

inefficient use of processor cache. Measurements taken with a sample workload using the CICS attachment facility shipped with CICS/ESA Version 4 within IBM show that performance starts to deteriorate when you have more than 100 threads per CICS region. ***Important:*** This value can differ in your environment. Many factors affect it, such as the version of MVS you are using. Just be aware that too many threads can cause performance to worsen.

5.5.1.3 PURGEC (for CICS/ESA Version 4 Only)

PURGEC allows you to specify the length of the protected thread purge cycle.

The CICS attachment facility has two types of threads, protected and unprotected.

- An unprotected thread is terminated immediately after the transaction is through with it (at syncpoint or EOT), unless another task is already waiting to use the thread.
- A protected thread remains for a certain length of time after the transaction is through with it, to increase the chances of thread reuse. The length of time that a protected thread remains is determined by the purge cycle time. The default purge cycle time is 30 seconds.

  It takes two purge cycles to terminate a protected thread; so if PURGEC is not specified, a protected thread remains between 30 and 60 seconds after the transaction is through with it.

  The purge cycle is 5 minutes long when the CICS attachment facility is started, and then it is fixed at PURGEC for the remainder of the time. The new RCT parameter PURGEC=(minutes,seconds) specifies the normal purge cycle length. For example, if you set PURGEC=(0,40), protected threads are purged 40-80 seconds after their last activity. The maximum specifiable length of a purge cycle is 59 minutes, 59 seconds. The minimum length is 30 seconds.

5.5.1.4 PLANI (for CICS/ESA Version 4 Only)

Use PLANI on the TYPE=INIT macro to specify a default plan name for any nondynamic selection entry in the RCT. Without PLANI, the plan name for any particular entry in the RCT is:

- The value for PLAN= in the TYPE=ENTRY form of the macro
- The value for TXID= on the TYPE=ENTRY form of the macro if PLAN= is not specified.

With PLANI, the plan name for an entry is:

- The value for PLAN= in the TYPE=ENTRY form of the macro
- The value of PLANI= in the TYPE=INIT form of the macro if PLAN= is not specified.

The PLANI option has no impact on entries using dynamic plan selection.

5.5.2 TYPE=COMD

This section deals with the parameters for the command thread.

The default parameters for the DB2 command thread are usually sufficient, but you can modify them. This is documented in the discussion of connecting the CICS attachment facility in the *IBM DATABASE 2 Administration Guide*.

Default values for the command thread are defined in the DSNCRCT member in the DSN230.DSNMACS, the DSN310.SDSNMACS, or the CICS410.SDFHMAC library. Do not modify this member, because of the possibility of IBM maintenance of the member. Instead of modifying the macro library, you should explicitly define the command thread in your RCT. You may want to change the TYPE=COMD statement to change the AUTH parameter to reflect the need to give different people different responsibilities in operating the DB2 system. A more detailed explanation of this is in section 4.4, “DB2 Security” on page 65. We highly recommend that you continue to use the default values THRDM=1, THDRA=1, THDRS=1, and TWAIT=POOL, because command threads are normally very low use threads. Defining more than one command thread can waste resources.

We recommend that you specify ROLBE=NO for command threads, because rollback is not used for DB2 commands.

The TXID parameter should always specify only DSNC. Even if using more than one command thread transaction, you need to specify only DSNC.

5.5.3 TYPE=POOL and TYPE=ENTRY

In this section we discuss the most important RCT parameters for the pool and entry threads.

5.5.3.1 DPMODE

This parameter controls the priority of the thread TCB. Most of the processor time used on the DB2 requests in the thread is controlled by this TCB. We recommend that you specify DPMODE=HIGH for high-priority and high-volume transactions. The goal is to execute these transactions quickly, to move them out of CICS to save virtual storage, and to move them out of DB2 to release the locks and avoid deadlocks with other transactions or timeouts.

However, if any SQL calls in your programs do massive amounts of work in DB2, then this thread TCB could monopolize the processor, and the CICS main TCB may not be dispatched. In this case, use DPMODE=LOW or DPMODE=EQ. If you are using only static SQL, this is not likely to occur in a *controlled environment* and you can specify DPMODE=HIGH. Sophisticated use of dynamic SQL can lead to situations in which you should not specify HIGH.

Note that when you are using a multiprocessor, the CICS main TCB and a thread TCB can be dispatched simultaneously and be executed in parallel in different processors. With these processors it can be acceptable to use DPMODE=HIGH, even if there is a risk of monopolizing a processor.

5.5.3.2 PLAN

PLAN specifies the name of the plan allocated to transactions defined in the TXID parameter. For the TYPE=POOL threads, the plan specified is used by only those transactions not explicitly defined in a TYPE=ENTRY statement. Transactions overflowing to the pool from an entry thread use the plan name specified on their TYPE=ENTRY statement. The TXID parameter value is the default plan name if only one transaction code is specified for the TXID parameter.

5.5.3.3 PLNEXIT

If this option is set to NO, then the CICS attachment facility obtains the plan name for the transaction IDs named in this TYPE=ENTRY or TYPE=POOL statement as described in section 5.5.1.4, “PLANI (for CICS/ESA Version 4 Only)” on page 86.

If this option is set to YES, then the CICS attachment facility uses the dynamic plan exit specified on the option PLNPGME= to obtain the plan name for the transaction IDs named in this TYPE=ENTRY or TYPE=POOL statement. You should not code PLAN= if you specified PLNEXIT=YES. The CICS attachment facility dynamically allocates the plan name (using the dynamic plan exit) when the first SQL statement is processed by the application program. If the transaction is using a pool thread, then this process of dynamically allocating a plan name is repeated for the first SQL statement after an EXEC CICS SYNCPOINT.

5.5.3.4 AUTH

AUTH specifies the type of authorization ID to use when passing an SQL request to DB2. For a description of the authorization options, see section 4.4, “DB2 Security” on page 65.

Note that there can be significant performance implications involved in the DB2 authorization checking. DB2 checks authorization when a thread is started and when a new authorization ID reuses an existing thread. Also see section 5.5.4.1, “Security” on page 93.

5.5.3.5 ROLBE

This parameter controls whether the CICS attachment facility requests CICS to issue a SYNCPOINT ROLLBACK if the transaction IDs named in this TYPE=ENTRY or TYPE=POOL statement is the victim of an IRLM-detected DEADLOCK or TIMEOUT.

If you specify ROLBE=YES, CICS rolls back all DB2 work and all other CICS-protected resources. The application program receives an SQL return code of -911 and can issue a message to the user and perform its own error processing. If this is a deadlock situation, then the other transaction can now continue.

In some cases, you can prepare the application code to either restart the transaction or to start the application program from the beginning again. In most cases, however, this is not possible because the original input can be lost. Also the -911 can be received in a module invoked by a CICS transfer control (XCTL) command, which can prevent the application program from finding where to start again.

We recommend in general that you specify ROLBE=YES.

One disadvantage of specifying ROLBE=NO is that DB2 rolls back only the incomplete SQL call that was involved in the deadlock or timeout. This avoids half-completed DB2 updates. CICS does not back out any resources. The application program receives an SQL code -913 and is responsible for taking further action. The normal case is either to issue a EXEC CICS SYNCPOINT ROLLBACK or to abend. All updates are then backed out and DB2 locks obtained in earlier SQL statements are released. The other transaction involved in the deadlock can continue. We recommend that your application program should issue an EXEC CICS SYNCPOINT ROLLBACK or abend by default when it receives a -913 return code.

Another use of ROLBE=NO is to give the application program the chance to try to continue without having other resources backed out. Two examples of this are:

- ***Timeout***. By analyzing the reason code in the SQL communication area (SQLCA), you can determine if the -913 resulted from a timeout. If so, and the DB2 resources you want to access are likely to be released soon, then your application program could reexecute the SQL statement a limited number of times until successful.

  Your application program can issue a CICS EXEC CICS DELAY to delay the execution of the next loop cycle.

- ***Single statement deadlock***. If the deadlock received is caused only by the last SQL statement in this program (and one or more SQL statements in another program) then it can be useful to execute the last SQL statement again. A typical example of this is an INSERT into a table with indexes.

A major disadvantage of using these techniques is that the program can hold other locks from previous SQL calls. These locks can now cause other transactions to time out or to go into another deadlock. In any of these cases, however, note that ROLBE=NO is specified for the whole plan allocated to the transactions in this group of threads. It means that all program modules having a DBRM included in the plan must be prepared to handle a -913 correctly in all SQL calls. If this is not the case, you should specify ROLBE=YES.

If you are using CICS distributed program link (DPL), you *must* specify ROLBE=NO for programs that do not specify SYNCONRETURN.

5.5.3.6 THRDM

This value specifies the maximum value to which THRDA can be set. It cannot be changed dynamically.

5.5.3.7 THRDA

This value specifies the current maximum number of active threads for this group of transactions. You can change THRDA dynamically using the DSNC MODIFY command. THRDA cannot exceed the value of THRDM.

THRDA can be increased and decreased dynamically for both pool threads and entry threads.

Note: If you decrease THRDA dynamically, the decrease is delayed until a thread purge takes place unless you have applied APAR PN70588 to your system. When you assemble your RCT, a macro check is done to ensure that THRDS is lower than or equal to THRDA. If THRDA=0 and TWAIT=POOL are specified, the requests for this entry are forced to the pool.

THRDA for the pool must be at least 3.

5.5.3.8 THRDS

This value defines the number of MVS TCBs to be attached when the CICS attachment facility is initiated. THRDS also defines the number of protected threads for this group of transactions.

If THRDS is specified for a thread, the thread is not terminated until an average of 45 seconds after a transaction uses it. In CICS/ESA Version 4, you can change this time by using the PURGEC parameter in the TYPE=INIT statement. However, if you do not code this parameter, the thread is terminated in an average of 45 seconds, as before.

If the thread is unprotected, it is terminated immediately after the end of the transaction, unless another transaction is queued for the thread.

Protected threads can be used to improve performance. Note that protected threads can be active for a long period, depending on the activity for the corresponding transactions. This can affect operation. If you try to bind a plan with the same name as the plan allocated to a thread, the BIND process can time out. The reason is that it is unacceptable to have one version of the plan active in the EDM-pool, and at the same time have another version of the plan physically placed in the SKCT data set. You can develop operational procedures to handle this situation. This is one reason for not using protected threads in a test CICS system.

A macro check is done at assembly time to check that the sum of THRDS values is less than or equal to THRDMAX.

Try to avoid a large number for THRDS. The CICS attachment facility attaches one TCB for each THRDS during initialization. This can result in a large number of TCBs in the CICS address space, causing more TCB dispatching time and unused TCBs that monopolize virtual storage resources.

You can review your shutdown statistics and check how many times a transaction waited for a thread (W/P). If this value is 0, then you could review your entry and check if you can decrease THRDS until values begin to appear in W/P. You can also merge plans of transactions not used very frequently and combine them into one RCT entry.

For an overview of recommendations regarding the THRDx and TWAIT values, see Figure 50 on page 91.

5.5.3.9 TWAIT

You can specify TWAIT=POOL, TWAIT=YES, or TWAIT=NO. If you specify TWAIT=POOL, a transaction associated with a TYPE=ENTRY statement uses a pool thread if all the entry threads for its group are being used. If you specify TWAIT=YES, the transaction waits until a thread becomes available. If you specify TWAIT=NO and no thread is available, the transaction abends.

Specify TWAIT=POOL for entry thread transactions that are allowed to overflow to the pool. You must specify TWAIT=POOL for entry threads with THRDA=0, to force the transactions to the pool. If you specify TWAIT=POOL for high volume transactions, the pool can be monopolized by these transaction types.

If you specify TWAIT=NO and no thread is available, the transactions abend. This is also the case for the pool. That means that overflow transactions may also be abended, if the pool is specified with TWAIT=NO.

You can use TWAIT=YES to limit the number of transactions using a particular plan concurrently. We recommend that you use the CICS facilities of transaction classes rather than TWAIT=YES to control concurrency. Set the maximum number of transactions in your class to THRDA plus one.

Transaction Characteristics	THRDM	THRDA	THRDS	TWAIT
Entry threads				
High-volume transactions	n	n	n	YES/POOL
Low-volume, high-priority transactions	1	1	0	POOL
Controlled transactions	n	n	0/n	YES
Serialized transactions	1	1	0/1	YES
Low-volume transactions	0	0	0	POOL
The POOL thread	>2	>2		YES

Figure 50. Recommendations for Specifying RCT Parameters

5.5.3.10 TXID

The order in which the transactions are specified is insignificant for the entry threads. If you use the DSNC MODI command to change the value of THRDA, any of the transactions specified in TXID can be referenced. Figure 68 on page 131 gives an example of the DSNC MODI command.

For pool threads, only the first specified transaction is accepted as a target for the DSNC MODI command.

The THRDA value for the command threads cannot be changed with the DSNC MODI command.

Note that when displaying the DB2 statistics with the DSNC -DISP command, the transaction *first defined* in the TXID parameter is the transaction displayed for the thread.

5.5.3.11 TOKENE

Because CICS produces accounting records on a transaction basis, and DB2 produces accounting records on a thread basis, it can be difficult to correlate the two. The TOKENE parameter gives you the option of producing a DB2 accounting record for each transaction, even for transactions that are reusing threads.

If you use TOKENE, it is possible to get more than one accounting record for a transaction. If your transaction is terminal-oriented, it releases the thread if you use an EXEC CICS SYNCPOINT command (unless, for example, you have an open cursor with hold). In this case a second accounting record is created if your transaction starts a second LUW containing SQL requests.

If you use TOKENE, it slightly increases the overhead of the SQL statement.

The advantage of using TOKENE is that the LU6.2 token, generated by CICS with every transaction, is passed to DB2 and included in the DB2 accounting record, making correlation between them easier.

5.5.3.12 TXIDSO (for CICS/ESA Version 4 Only)

The TXIDSO option of the RCT allows you to suppress some sign-ons during thread reuse, and therefore prevent extraneous accounting records from being written.

With the default of TXIDSO=YES, the CICS attachment facility determines whether a new transaction can reuse an existing thread in this way:

- If the plan name, authorization ID, and transaction ID are the same as the last transaction that used the thread, and if TOKENE is set to NO, then the transaction reuses the thread without any sign-on.
- If the plan name is the same as that of the transaction that last used the thread, and
  - The authorization ID changed, or
  - TOKENE is YES, or
  - The transaction ID changed

  then the thread is reused following a sign-on.
- If the plan name changed, the thread is terminated and re-created.

If you specify TXIDSO=NO, the logic changes to:

- If the plan name and authorization ID are the same as the last transaction that used the thread, and if TOKENE is set to NO, then the transaction reuses the thread without any sign-on.
- If the plan name is the same as that of the transaction that last used the thread, and
  - The authorization ID changed, or
  - TOKENE is YES

  then the thread is reused following a sign-on.
- If the plan name changed, the thread is terminated and re-created.

The TXIDSO option affects only pool threads and those RCT entry threads with multiple transaction IDs in one entry, for example TXID=(XC05,XC07). This is because the attach only checks for thread reuse within an entry.

Note: TXIDSO has no effect on transactions that specify TOKENE=YES.

The TXIDSO option is new with CICS/ESA Version 4. The default value is YES. Other releases of CICS behave as if TXIDSO were always YES.

5.5.4 Security and Grouping of Transactions

This section discusses security and grouping of transactions.

5.5.4.1 Security

Authorization checks take place when a thread is created and when a thread is reused with a new user. Avoiding the overhead in the security check is part of the performance advantage of using protected threads. This means that from a performance viewpoint all transactions defined in an RCT entry with THRDS>0 should use the same authorization ID. At least they should avoid specifying TERM, USER, and USERID for the AUTH parameter, because these values change often.

5.5.4.2 Grouping Transactions

Several transactions can be specified in the same RCT entry. They must all use the same plan. Each low-volume transaction can have its own entry in the RCT. In this case thread reuse is not likely and you should specify THRDS=0. An alternative is to group a number of low-volume transactions in the same RCT entry and specify THRDS=n. This gives better thread utilization and less overhead.

Note that the plan specified for this entry must include the DBRMs for all modules that any of the transactions can execute.

5.6 Creating, Using, and Terminating Threads

In this section we deal with the processes involved in creating, using, and terminating threads, and attaching thread TCBs, and the implications of these on specifying RCT parameters. We also discuss overflow to the pool. We look closely at thread creation and termination for:

- Protected entry threads
- High-priority unprotected entry threads
- Low-priority unprotected entry threads
- Pool threads.

You define the relationship between CICS transactions and DB2 plans in the RCT by specifying the TYPE=POOL and TYPE=ENTRY entries in the RCT macro.

The general rules that apply to thread creation, use, and termination are:

- When the CICS attachment facility is started, the number of TCBs attached for each RCT entry is the value specified for THRDS.
- Before an SQL request can be passed to DB2, a TCB and a thread must be available for the transaction.
- Once attached, a TCB is normally available until the CICS attachment facility is stopped. However, if the number of thread TCBs exceeds THRDMAX-2, the CICS attachment facility may detach a TCB and the corresponding thread.
- When a thread is created and when another transaction with a new authorization ID is reusing a thread, DB2 makes an authorization check of the new authorization ID.
- A terminal-oriented transaction usually releases the thread at SYNCPOINT and EOT. The thread is *not* released at SYNCPOINT if held cursors are open or any modifiable special registers are not in their initial state.
- A non-terminal-oriented transaction releases the thread at EOT only.

- When a transaction releases a thread, the thread can be reused by another transaction specifying the same plan and defined in the same RCT entry. Pool threads can be reused by any waiting (queued) transaction specifying the same plan and using a pool thread.
- An unprotected thread is terminated immediately when it is released unless another transaction is waiting (queued) for the thread.
- The protected thread is terminated if not used for two consecutive periods of 30 seconds. This averages about 45 seconds. With CICS/ESA Version 4 this value can be modified by the PURGEC parameter in the RCT TYPE=INIT entry.
- The TWAIT parameter defines whether the requests for a thread should be queued, abended, or overflowed to the pool in case of entry thread or command thread shortage.

In the following sections these rules are applied to a number of different types of RCT specifications. A more detailed explanation of thread creation, use, and termination is given for each of the above-mentioned thread types. Figure 51 on page 95 shows the correlation between THRDM, THRDA, THRDS, and the TCBs for various thread types.

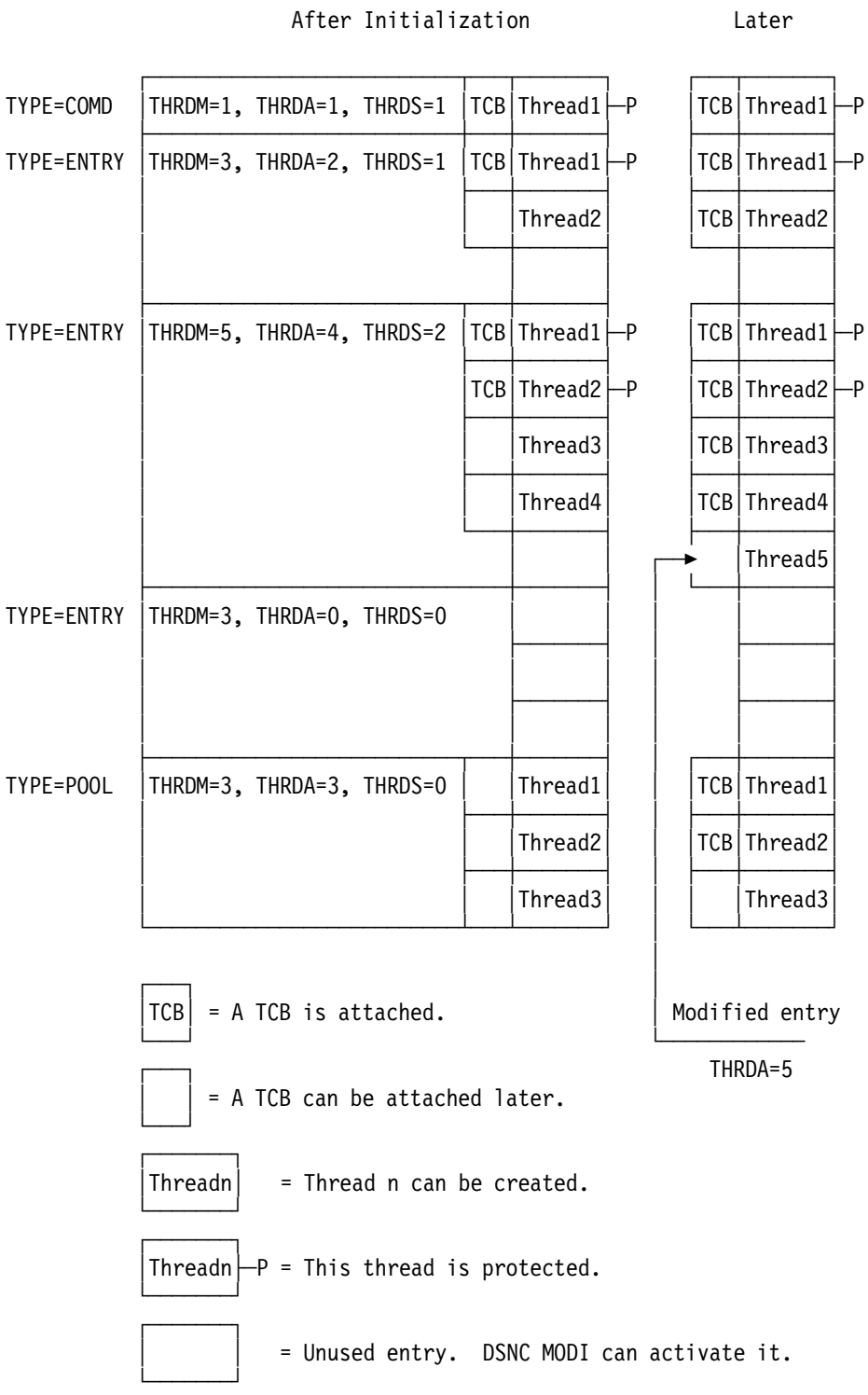


Figure 51. RCT-Related TCBs and Threads. Correlation between RCT specifications, threads, and TCBs after CICS attachment facility initialization and in the middle of production.

A special case, not shown in Figure 51, is for a protected thread to lose its TCB. If the number of thread TCBs comes to within the THRDMAX-2 limit, one or more

thread TCBs can be detached to allow other TCBs to be attached; this includes TCBs for protected entry threads.

5.6.1 Protected Entry Threads

These threads are defined with the following key parameters:

TYPE=ENTRY with THRDS=n and THRDA=n

Protected entry threads are recommended for:

- High-volume transactions of any type
- Terminal-oriented transactions with many commits.

Protected entry threads are created, used, and terminated as follows:

TCB attach A total of n thread TCBs are attached when the CICS attachment facility is started. These TCBs normally remain available until the CICS attachment facility is stopped[2].

Thread creation A thread is created only if an existing thread is not available.

If no thread is available, but an unused TCB exists for this RCT entry, a new thread is created and related to the TCB.

When THRDA is greater than THRDS and the limit for THRDS has been reached, both a new TCB and a new thread are created. This thread is an unprotected entry thread. It is terminated as soon as it is released, unless it has a transaction queued for it. The new TCB continues to be available for this RCT entry.

Thread termination Any thread is terminated if it is unused for two consecutive 30 second periods or PURGEC value (CICS/ESA Version 4 only).

Thread reuse Other transactions specified in the TXID of the same RCT entry may reuse a thread, if it is available. This is likely because the threads remain active for about 45 seconds after being released. In CICS/ESA Version 4 this value depends on the setup of the PURGEC parameter.

Overflow to pool If TWAIT=POOL is specified, requests for threads are transferred to the pool when the value of THRDA is passed. A transaction is then under the control of the DPMODE, THRDA, and TWAIT parameters for the pool. The transaction keeps the PLAN and the AUTH values you specified for the entry thread.

---

[2] If the number of thread TCBs is within the THRDMAX-2 limit, one or more thread TCBs can be detached to allow other TCBs to be attached.

5.6.2 High-Priority Unprotected Entry Threads

These threads are defined with the following key parameters:

TYPE=ENTRY with THRDS=0 and THRDA=n

This is the recommended type of definition for:

- High-priority transactions, but with a volume so low that a protected thread cannot be used.
- Limited concurrency transactions.

You could use a protected thread for limited concurrency transactions, if the transaction rate makes it possible to reuse the thread.

High-priority unprotected entry threads are created, used, and terminated as follows:

TCB attach No thread TCBs are attached when the CICS attachment facility is started.

A TCB is attached only if needed by a thread, and normally remains available until the CICS attachment facility is stopped.

Thread creation A thread is created only when needed by a transaction.

If no thread is available, but an unused TCB exists for this RCT entry, a new thread is created and related to the TCB.

If the limit for THRDA was not reached and a TCB does not exist, both a new TCB and a new thread are created.

Thread termination The thread is terminated immediately after it is released, unless it has a transaction queued for it.

Thread reuse Other transactions specified in the TXID of the same RCT entry can reuse a thread, if it is available. This is not likely, though, because the thread is terminated after use.

Overflow to pool If TWAIT=POOL is specified, requests for threads are transferred to the pool when the value of THRDA is passed. A transaction is then under the control of the DPMODE, THRDA, and TWAIT parameters for the pool. The transaction keeps the PLAN and the AUTH values you specified for the entry thread.

Note that you should not allow overflow to the pool for limited concurrency transactions.

5.6.3 Low-Priority Unprotected Entry Threads

These threads are defined with the following key parameters:

TYPE=ENTRY with THRDS=0, THRDA=0, and TWAIT=POOL

This is the recommended type of definition for transactions with low volume and low priority. All transactions are forced to use pool threads.

Low-priority unprotected entry threads are created, used, and terminated as follows:

TCB attach No thread TCB is ever attached for this thread definition because THRDA=0. A pool thread TCB is used. All activity related to this entry definition is forced to a thread and a TCB in the pool. A transaction is then under the control of the DPMODE, THRDA, and TWAIT parameters for the pool. The transaction keeps the PLAN and the AUTH values you specified for the entry thread.

Thread creation A thread is created in the pool when needed by a transaction unless the THRDA value for the pool was reached.

Thread termination The thread is terminated when it is released, unless it has a transaction queued for it.

Thread reuse Other transactions using the same plan can reuse the thread, when it becomes available.

5.6.4 Pool Threads

These threads are defined with the following key parameters:

TYPE=POOL with THRDA=n and THRDA=n

There are three cases when a pool thread can be used:

- A transaction is not defined in any TYPE=ENTRY statement, but issues SQL requests. This transaction defaults to the pool and uses the plan specified for the pool.
- A transaction is defined in a TYPE=ENTRY statement, but is forced to the pool because the TYPE=ENTRY statement specifies THRDA=0 and TWAIT=POOL. This transaction uses the plan specified in the entry statement.
- A transaction is defined in a TYPE=ENTRY statement, but overflows to the pool (TWAIT=POOL) when the THRDA value is passed. This transaction uses the plan specified in the entry statement.

Pool threads are always unprotected threads, even if you specify a THRDS value greater than zero.

Low-priority unprotected entry threads are created, used, and terminated as follows:

TCB attach The THRDS parameter is ignored for the pool. TCBs are attached when needed for pool threads. These TCBs normally remain available until the CICS attachment facility is stopped.

Thread creation A thread is created only when needed by a transaction.

Thread termination The thread is terminated immediately after it is released, unless it has a transaction queued for it.

Thread reuse Other transactions using the same plan can reuse a thread when it becomes available. In the pool this happens only if there is a queue for threads and the first transaction in the queue requests the same plan used by the thread being released.

5.7 Sample RCT Specification

In this section we give an example of how you might initially specify the RCT parameters when you have some knowledge of the transactions.

The recommended bind options for the corresponding plan are included in the examples. Requirements other than those discussed can lead to a different specification. For example, we assume that the transactions are running on an unconstrained system. The principles used in the examples should still be valid.

The sample RCT is defined to reflect the situation described in Figure 52.

TRANS ID	Transaction Rate per Minute Avg.	Max.	Expected Task Duration (Seconds)	Application Priority	Concurrency Requirement	Plan to Use	Remarks
T001	300	600	1.0	high	max	P1	Well tuned
T002	60	90	0.5	high	max	P2	High volume
T022	30	30	2.0	high	max	P2	High volume
T003	0	3	1.0	low	max	P3	
T004	1	15	2.0	low	1 only	P4	
T005	4	5	9.0	low	2 only	P5	Not tuned
T006	1	3	2.0		max	P6	Small plan
T007	1	3	3.0		max	P7	Small plan
T008	1	3	4.0		max	P8	Small plan
T009	1	1	1.5	High	max	P9	Must run fast
B001	0	1	600.0	low		PB	Non-terminal
TB01	0	1	600.0	low	1 only	PT	Many commits

Figure 52. Sample Transactions. The figure shows the initial expectations of transaction characteristics.

The following example explains the contents of the table: We expect transaction T001 to execute 300 times a minute on average. In peak hours, it executes 600 times a minute. We expect that the average response times for T001 in the peak hour is about 1.0 second. It has high priority. We do not expect any concurrency problems. It uses the DB2 plan P1 and is well tuned.

The following general assumptions apply to all the transaction types:

- All transactions issue SQL requests.
- The first SQL call is at the beginning of the transaction, which means that the thread will be busy approximately the same amount of time that the task is alive.
- RCT specifications mainly reflect peak hour requirements.
- We specified ROLBI=YES in the TYPE=INIT statement.
- No transaction executes LOCK TABLE statements.

We discuss the transaction examples listed in Figure 52 under the headings shown in Figure 53 on page 100 and in the subsequent subsections.

Transaction Characteristics	Transaction Examples
High volume, high priority	T001, T002/22,
Low volume, high priority	T009, B001
Low volume	T006/7/8
Limited concurrency	TB01, T004, T005
Low volume, low priority	T003
The Pool	The Pool

Figure 53. Classification of Examples. The table shows under which heading the examples are discussed.

5.7.1 High-Volume, High-Priority Transactions

T001 First calculate the nominal number of 100% busy threads.

In the peak period there are 600/60 = 10 transactions/second (TPS). Each transaction is expected to occupy a thread for one second. That gives 10 TPS x 1 second = 10 concurrently busy threads on average in the peak hour.

It is reasonable to expect some variations during the peak hour. We allow for a 30% variation, specifying a total of 13 threads initially.

THRDA should be set to 13. THRDS should be between 10 and 13. Using one TCB more or less does not really matter for this important transaction. In this case, we decide to use 13 protected threads.

We do not use TWAIT=POOL in this case, due to the risk of monopolizing the pool. If for example a deadlock occurs between two T001 transactions, they occupy two threads (typically for 15-60 seconds, depending on the IRLM deadlock detection cycle parameter). A cascade effect could occur, which could lead to a blockade of the pool, if transactions were allowed to use the pool. It would probably be better to increase THRDA instead. The THRDA value should be high enough to handle:

- Unexpected response times
- Increased transaction rate
- One or more threads having transactions involved in deadlock or timeout.

DPMODE is defined as HIGH. These transactions are well tuned.

AUTH=(SIGNID,*,*) is used because the overhead in DB2 for authorization checks of a new user is undesirable. We could use the TXID (T001) or a character string instead of SIGNID. This also gives an unchanged authorization ID from transaction to transaction.

We specify that all TS locks referenced in the plan are acquired when the thread is started and released at thread termination to obtain high performance.

The complete definition for T001 is:

```
DSNCRCT TYPE=ENTRY,
AUTH = (SIGNID,*,*),
DPMODE = HIGH,
PLAN = P1,
THRDM = 13,
THRDA = 13,
THRDS = 12,
TWAIT = YES,
TXID = (T001)
```

BIND options: ACQUIRE(ALLOCATE), RELEASE(DEALLOCATE)

T002/22 These two transactions use the same plan, P2. The same RCT entry is used for both transactions. The nominal number of busy threads is calculated as before. For T002 and T022, it is 0.75 and 1.0, or a total of 1.75 threads. In this case two protected threads are used and THRDA = 3 is specified to allow for peak periods.

Plan P2 refers to many table spaces. We use the bind options:

```
ACQUIRE(ALLOCATE) and
RELEASE(DEALLOCATE)
```

This avoids acquiring or releasing the TS locks for each transaction. If the third thread is unprotected, it uses additional resources for acquiring or releasing all possible TS locks for each transaction using this thread; so we modify our decision on the number of protected threads and use three. Note that we expect the third thread to be used many times. On average 1.75 threads are busy in the peak hour. There are relatively big differences in response times for the transactions, and even small variations from the average can require use of the third thread.

The complete definition for T002/22 is:

```
DSNCRCT TYPE=ENTRY,
AUTH = (SIGNID,*,*),
DPMODE = HIGH,
PLAN = P2,
THRDM = 3,
THRDA = 3,
THRDS = 3,
TWAIT = YES,
TXID = (T002,T022)
```

BIND options: ACQUIRE(ALLOCATE), RELEASE(DEALLOCATE)

5.7.2 Low-Volume, High-Priority Transactions

T009 This transaction must run fast. It is executed only once a minute. This does not allow it to keep a thread busy (unless we use the PURGEC option of the TYPE=INIT statement to increase the purge cycle time, which may not be desirable). We eliminate the possibility of being queued up in the pool by using one thread. Overflow transactions are allowed in case of concurrency.

Only the TS locks necessary for the specific execution of the transaction should be acquired (to minimize the costs of thread

creation), and there is only one LUW in the transaction; so we use the following bind options.

ACQUIRE(USE)and
RELEASE(COMMIT)[3]

The complete definition for T009 is:

```
DSNCRCT TYPE=ENTRY,
AUTH = (SIGNID,*,*),
DPMODE = HIGH,
PLAN = P9,
THRDM = 1,
THRDA = 1,
THRDS = 0,
TWAIT = POOL,
TXID = (T009)
```

BIND options: ACQUIRE(USE), RELEASE(COMMIT)

B001 This is a non-terminal-attached transaction in CICS running for about 15 minutes several times a day. It often issues SYNCPOINTs (about five times a second) to commit resources and to release locks in DB2. It does not release the thread at syncpoint, because it is non-terminal attached. In this case it should be executed in the pool to avoid having a TCB attached and unused after the first execution of the program.

The complete definition for B001 is:

```
DSNCRCT TYPE=ENTRY,
AUTH = (SIGNID,*,*),
DPMODE = LOW,
PLAN = PB,
THRDM = 0,
THRDA = 0,
THRDS = 0,
TWAIT = POOL,
TXID = (B001)
```

BIND options: ACQUIRE(ALLOCATE), RELEASE(DEALLOCATE)

5.7.3 Low-Volume Transactions

T006/7/8 These transactions are spread throughout the day. None of them makes efficient use of a protected thread alone. The thread is likely to be terminated before the next transaction arrives. These transactions are grouped in the same RCT entry and a protected thread is defined. That requires one common plan. This plan must be bound to include the DBRMs for all modules in the three transactions. The new plan is PN. Overflow to the pool is allowed.

The complete definition for T006/7/8 is:

[3] If this transaction issues syncpoints, the RELEASE(COMMIT) should be changed to RELEASE(DEALLOCATE) to avoid taking the same TS locks more than once. In this case we would also use a protected thread to be able to reuse the thread between the LUWs in the transaction. Both changes must be made to obtain the advantage.

```
DSNCRCT TYPE=ENTRY,
AUTH = (SIGNID,*,*),
DPMODE = EQ,
PLAN = PN,
THRDM = 1,
THRDA = 1,
THRDS = 1,
TWAIT = POOL,
TXID = (T006,T007,T008)
```

BIND options: ACQUIRE(ALLOCATE), RELEASE(DEALLOCATE)

5.7.4 Limited Concurrency Transactions

TB01 This is the same transaction as B001, but terminal attached. It releases the thread at syncpoint because it is terminal attached. In this case it should use a protected thread. If the thread is unprotected, additional resources are used at each syncpoint to:

- Terminate and start the thread.
- Release and acquire TS locks.

The cost of this is greater than the cost of having an unused TCB in the system.

If two of these transactions try to execute in parallel, the second one should abend.

The complete definition for TB01 is:

```
DSNCRCT TYPE=ENTRY,
AUTH = (SIGNID,*,*),
DPMODE = LOW,
PLAN = PT,
THRDM = 1,
THRDA = 1,
THRDS = 1,
TWAIT = NO,
TXID = (TB01)
```

BIND options: ACQUIRE(ALLOCATE), RELEASE(DEALLOCATE)

T004 This transaction must be restricted to run without concurrency. It will keep 0.5 thread busy to allow using a single thread. The serialization is done by allowing only one thread for the transaction and not allowing overflow to the pool. CICS transaction classes could have been used to serialize the transactions.

This transaction is able to take advantage of a protected thread in a peak hour, because it executes more than one to two times in a minute.

We give this transaction a high priority to compensate for the serialization, because we want to avoid other transactions queueing up behind it.

The complete definition for T004 is:

```
DSNCRCT TYPE=ENTRY,
AUTH = (SIGNID,*,*),
```

```
DPMODE = HIGH,
PLAN = P4,
THRDM = 1,
THRDA = 1,
THRDS = 1,
TWAIT = YES,
TXID = (T004)
```

BIND options: ACQUIRE(ALLOCATE), RELEASE(DEALLOCATE)

T005 This transaction is not very well tuned and it is expected to run for about nine seconds, five times a minute. The transaction is not allowed to hang up other DB2 resources, in case the estimate is wrong. The concurrency is limited to two transactions at a time by specifying THRDA=2 and TWAIT=YES.

The transaction can also hang up CICS resources before being queued up waiting for a thread. To avoid this, CICS techniques should be used to limit the concurrency of the transaction in CICS.

The complete definition for T005 is:

```
DSNCRCT TYPE=ENTRY,
AUTH = (SIGNID,*,*),
DPMODE = LOW,
PLAN = P5,
THRDM = 2,
THRDA = 2,
THRDS = 1,
TWAIT = YES,
TXID = (T005)
```

BIND options: ACQUIRE(ALLOCATE), RELEASE(DEALLOCATE)

5.7.5 Low-Volume, Low-Priority Transactions

T003 This transaction is active only in the peak hour. It should go to the pool because it cannot use a TCB and a thread by itself.

We defined it in a separate entry thread with a specific plan. T003 could have executed directly in the pool. In that case it must have used the plan specified in the TYPE=POOL statement. No specification for T003 in the RCT was necessary in that case.

The complete definition for T003 is:

```
DSNCRCT TYPE=ENTRY,
AUTH = (SIGNID,*,*),
DPMODE = LOW, [4]
PLAN = P3,
THRDM = 0,
THRDA = 0,
THRDS = 0,
TWAIT = POOL,
TXID = (T003)
```

BIND options included: ACQUIRE(USE), RELEASE(COMMIT)

5.7.6 The POOL

We must also calculate the expected number of busy threads for the pool. This is the sum of the threads occupied by:

- Transactions forced to the pool with THRDA=0
- Overflow transactions
- Pool-defined transactions.

5.7.6.1 Transactions Forced to the Pool with THRDA=0

These transactions are T003 and B001.

T003 will occupy a thread for about three seconds each minute. B001 runs for about 15 minutes several times each day. When running, it occupies one thread.

5.7.6.2 Overflow Transactions

T006/7/8 occupies a thread in a total of 6+9+12 = 27 seconds per minute. T009 only seldom uses the pool.

5.7.6.3 Pool-Defined Transactions

In the examples no transactions were defined directly in the pool. In this group, transactions not defined in any entry at all can also run. They default to the pool and use the plan specified for the pool.

Assume, however, that *on average* one thread is busy with such transactions.

5.7.6.4 Total Pool Thread Busy Time

If the B001 transaction is running at the same time that the peak occurs for the other transactions, three pool threads are likely to give some queuing delays. This can happen although less than three threads are busy on average. We specify four pool threads. This also allows some pool threads to be busy with deadlock or timeout.

We may want to increase the THRDA value to more than the initial settings, therefore we set THRDM to seven. We can change THRDA value with the DSNC MODI command, but the DSNC MODI command must reference the first transaction specified in the TXID option. Therefore, we place a dummy transaction POOL in front of the TXID parameter. This allows us to reference that transaction when changing THRDA for the pool.

All transactions not defined in a TYPE=ENTRY statement must use the same plan. An example of this is shown below. Three transactions xxxx, yyyy, and zzzz are using the plan pppp. Other transactions with no RCT definitions can also be executed in the pool with the plan pppp.

The pool threads have DPMODE=EQ. Overflow transactions from entry threads also use the dispatching priority of the pool thread.

The complete definition for the pool is:

```
DSNCRCT TYPE=POOL,
AUTH = 'POOLONLY',
DPMODE = EQ,
```

[4] DPMODE = LOW has no meaning here because this transaction will use a TCB in the pool.

```
PLAN = pppp,
THRDM = 7,
THRDA = 4,
TWAIT = YES,
TXID = (POOL,xxxx,yyyy,zzzz)
```

5.7.7 THRDMAX

We can now calculate the THRDMAX parameter of the TYPE=INIT statement. The value defines the maximum number of threads that can be started between CICS and DB2.

We recommend specifying a value three greater than the sum of:

- THRDA for the command threads
- THRDA for all entry threads
- THRDA for the pool threads.

However, if THRDM is greater than THRDA for any thread, it is possible to dynamically increase the THRDA value during the production period. Unless dynamically increasing THRDA is under tight control, we recommend using the sum of the THRDM values. In our example the sum of THRDM is 29. THRDMAX should then be set to 29+3 = 32.

Chapter 6. Program Preparation and Testing

In this chapter we discuss program preparation and testing in a CICS-DB2 environment. The purpose of this chapter is to describe:

- Different possible environments where the programs could be prepared and tested
- The steps required for program preparation and testing
- The bind process and the consequences of different bind options
- DSNTIAC, the CICS SQLCA formatting routine
- Tools for program testing and debugging
- Considerations for moving an application from test to production.

6.1 The Test Environment

You can connect more than one CICS system to the same DB2 system. However, the CICS attachment facility does not allow you to connect one CICS system to more than one DB2 system at a time.

You can set up production and test environments with:

- A single CICS system connected to one DB2 system
- Two or more CICS systems for production and test, connected to the same DB2 system
- Two or more CICS systems, as above, connected to two or more different DB2 systems (one test and one production DB2 system).

The second alternative with just one DB2 system could be used for both test and production. Whether it is suitable depends on the development and production environments involved.

The third alternative with, for example, one test and one production DB2 system is the most flexible.

6.1.1 One CICS System

DB2 is designed to allow dynamic changes to its resources while it is active. CICS, with its resource definition online facility, allows programs, transactions, maps, and terminals to be dynamically added to a running system.

A concern in running both production and test in the same CICS system can be the RCT. New transactions cannot be added, since it is not possible to dynamically change the RCT. The alternative of stopping the connection and starting with an RCT that contains the new program cannot normally be permitted in a production environment.

For these reasons, we do not recommend using a single CICS system for both production and test.

6.1.2 Two CICS Systems

Running a test CICS system and a production CICS system separately allows you to start and stop the test connection without impacting production. When new programs are ready for testing, you can add entries for them to the RCT and reassemble the RCT. By stopping and then starting the attachment facility, you activate the new version of the RCT.

Two CICS subsystems can run with one or two DB2 systems. Where the CICS systems are attached to different DB2 systems:

- User data and the DB2 catalog are not shared. This is an advantage if you want to separate test data from production data.
- Wrong design or program errors in tested applications do not affect the performance in the production system.
- Authorization within the test system can be less strict because production data is not available. When two CICS systems are connected to the same DB2 system, authorization must be strictly controlled, both in terms of the functions and the data that are available to programmers.

The choice of one or two DB2 systems is dictated primarily by application demands.

6.2 Preparation Steps

The steps shown in Figure 54 on page 109 summarize how to prepare your program for execution after your application program design and coding is complete.

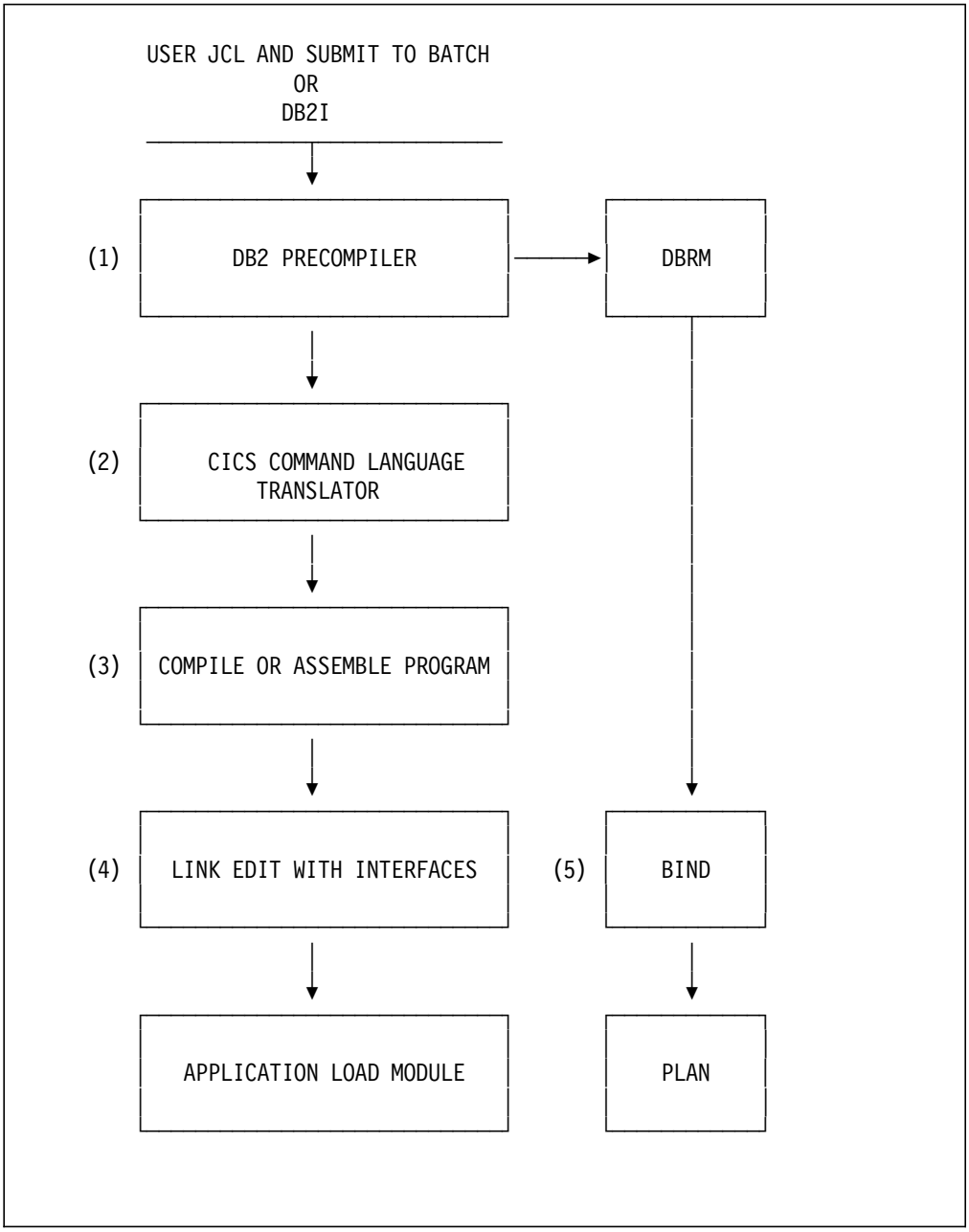


Figure 54. Steps to Prepare a CICS Application Program

Figure 54 shows that there are several steps in the process of preparing a CICS application program. When you prepare your own CICS application programs:

- You can run steps (1) and (2) in either sequence. However, we recommend the sequence shown because it is also used by the DB2I program preparation panels.
- If the source program is written in COBOL, you must specify a string delimiter that is the same for the DB2 precompiler, COBOL compiler, and CICS translator. The default delimiters are not compatible.
- If the source program is written in PL/I, the input to step (1) is the output from the PL/I macro phase (if used).
- The extra steps that are required for CICS application programs that access DB2 resources are DB2 precompile (1) and application program bind (5).

The DB2 precompiler builds a DBRM that contains information about each of the program's SQL statements. It also validates SQL statements in the program.

In the bind process, the DBRM produces an application plan that allows the program to access DB2 data at execution time.

A group of transactions specified in the same RCT entry must use the same application plan, that is, their DBRMs must be bound together.

- Both the appropriate CICS language interface module and the DB2 language interface module must be included in the link edit of the program. The CICS language interface *must* be included first in the load module (4).
- DB2 is required only for the bind process (5).

You can perform the program preparation shown in Figure 54 on page 109 using the DB2 Interactive Interface (DB2I) or by submitting your own JCL for batch execution.

- DB2 Interactive Interface (DB2I). DB2I provides panels to precompile, compile or assemble, and link-edit an application program and to bind the plan. For details about application program preparation, see *IBM DATABASE 2 Application Programming and SQL Guide*.
- User JCL submitted to batch execution. Members DSNTEJ5C and DSNTEJ5P in the library DSN230.DSNSAMP for DB2 2.3 or DSN310.SDSNSAMP for DB2 3.1 contain samples of the JCL required to prepare COBOL and PL/I programs for CICS. Section 2.3.7, "Installation Verification Procedure" on page 34 contains an inline JCL procedure that you can use to prepare a COBOL application program for execution, instead of using the DSNH CLIST.

If you perform this process while CICS is running, you may need to issue a CEMT NEWCOPY command to make the new version of the program known to CICS.

6.3 What to Bind after a Program Change

The example in Figure 55 on page 111 shows a CICS transaction, consisting of four program modules. It is not unusual that the number of modules is high in a real transaction. This section describes what you must do if one module is changed.

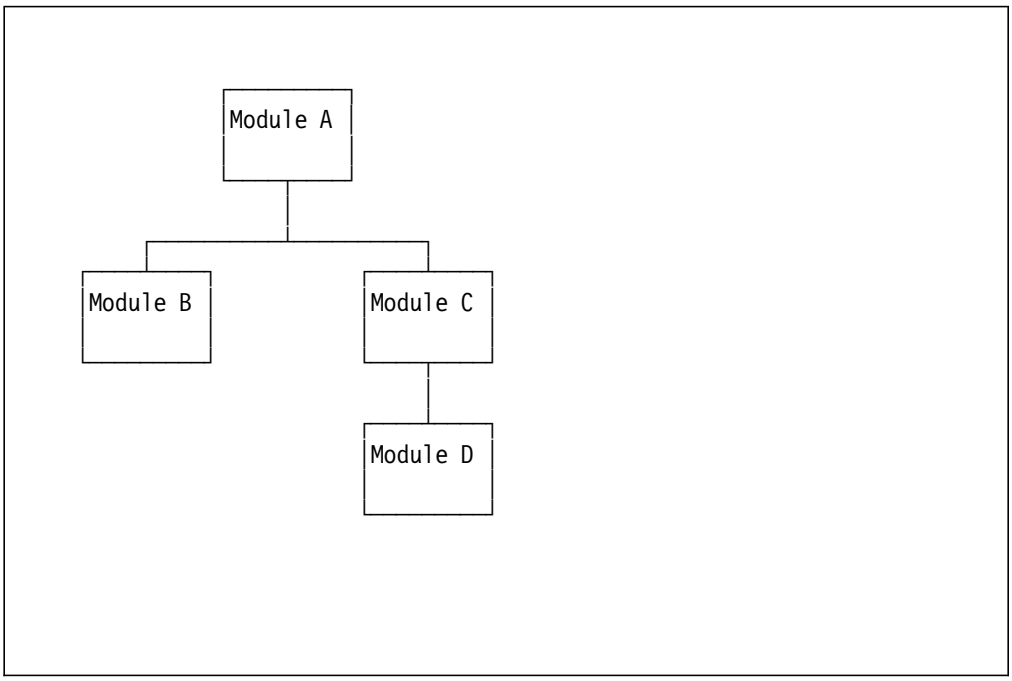


Figure 55. Application Consisting of Four Program Modules

Assuming that at least one SQL statement changed in program C, you must perform the following steps to prepare the program and to make the transaction executable again:

1. DB2 precompilation of the program
2. CICS command language translation
3. Compilation of host language source statements
4. Link-edit
5. Bind with DBRM names for all the programs specified, to get a new application plan. For the programs that were not changed, use their old DBRMs. Note that you *cannot* use the REBIND subcommand, because input to REBIND is the plan and *not* the DBRMs.

If program C is a commonly used module, you must BIND all application plans where *this program's DBRM* is included. If you use packages you only need to BIND the package for this DBRM.

Using packages simplifies the rebinding process. In fact, you could bind each separate DBRM as a package and include them in a package list. The package list can be included in a PLAN. You can use the BIND PACKAGE command to rebind the modified application instead of the BIND PLAN. This provides increased transaction availability and better performance, because the CICS attachment facility deals with PLANS and not with PACKAGES. See section 10.2.7, “Packages” on page 197 and *IBM DATABASE 2 Application Programming and SQL Guide* for details of packages implementation.

6.4 Bind Options

Almost all bind options are application dependent and should be taken into account during the application design. Procedures should be established for bind with different options, either manual or automatic. However, there is one bind option to consider here. It is the RETAIN option. RETAIN means that BIND and EXECUTE authorities from the old plan are not changed.

When the RETAIN option is not used, all authorities from earlier GRANTs are REVOKED. The user executing the BIND command becomes the creator of the plan, and all authorities must be reestablished by new GRANT commands.

This is why we recommend using the RETAIN option when binding your plans in the CICS environment.

6.5 CICS SQLCA Formatting Routine

DSNTIAR, the IBM supplied SQLCODE message formatting procedure, lets you send a sort of ″SQL messages online″ to your application.

With DB2 3.1, DSNTIAR was split into two front-end modules (DSNTIAC and DSNTIAR) and a run-time module (DSNTIA1). DSNTIAC is used for CICS applications and DSNTIAR for other DB2 interfaces. This change removes the need, previous to DB2 3.1, to relink-edit your application modules again every time a change is made to DSNTIAR, either by change of release or by applying maintenance.

The CICS front-end part, DSNTIAC, is supplied as a source member in DSN310.SDSNSAMP.

It is very important to have APAR PN45895 applied on your DB2 subsystem to improve performance and also to have DSNTIAC available in your DSN310.SDSNSAMP library. Before this APAR, DSNTIAC was named DSNCIAR. DSNCIAR was not very efficient in managing CICS storage, as it requested a GETMAIN for loading DSNTIA1 and passing control whether the SQL return code was zero or not. So, if you have applications that have DSNTIAR link-edited in your application module, and you are running a DB2 V3 subsystem, then you should consider the possibility of using DSNTIAC instead. In that case, you must relink-edit your modules, either using another library or renaming the modules for fall back.

Future relink-edits should not be required for future changes to SQLCODE messages for maintenance and for migrations from DB2 V3. APAR PN45895 should be applied to make DSNTIAC available and to minimize performance degradations.

6.6 Program Testing and Debugging

The tools that can be used in testing and debugging a CICS application program that accesses DB2 are those normally used in a CICS environment. These include:

- The execution diagnostic facility (EDF)
- The CICS auxiliary trace
- Transaction dumps.

The SQL Processor Using File Input (SPUFI) can also be useful in validating SQL statements and displaying the test tables to verify that the program functions correctly.

6.6.1 The Execution Diagnostic Facility

An SQL statement is traced in the same way as other CICS commands.

The CICS-DB2 EDF support helps you to develop and debug CICS-DB2 applications. In CICS/MVS Version 2, EDF did not display SQL statements, and the address of the argument was passed in the EDF panel. With CICS/ESA, SQL statements can be displayed in EDF. Table 7 shows the level of SQL statement display.

Table 7. EDF Support in CICS and DB2 Releases

	CICS/ESA Version 3	**CICS/ESA Version 4**
DB2 2.3	YES*	YES
DB2 3.1	YES	YES
Note: For DB2 2.3 with CICS/ESA Version 3, no authorization ID is displayed.		

In EDF mode, the CICS attachment facility:

- Stops on every EXEC SQL command and deciphers the SQL statement, showing it in a field on the panel
- Shows the results before and after SQL statements are processed
- Displays:
  - The type of SQL statement
  - Any input and output variables
  - The contents of the SQLCA
  - Primary and secondary authorization IDs. (This helps to diagnose SQLCODE -922).

An EDF panel displays a maximum of 55 variables, which is about 10 screens. Each EDF SQL session requires 12K of CICS temporary storage, which is freed when exiting EDF.

You should define all CICS attachment facility programs (DSNCxxxx or DSN2xxxx) with EDF(NO) or CEDF(NO) in your CICS CSD.

Figure 56 on page 114 and Figure 57 on page 114 show CICS EDF panels for SQL statements.

```
TRANSACTION: XCO5 PROGRAM: TESTCO5 TASK:0000097 APPLID: CICS41 DISPLAY:00
 STATUS:  ABOUT TO EXECUTE COMMAND

 EXEC SQL OPEN
  DBRM=TESTCO5,  STMT=00221, SECT=00001









 OFFSET:X'001692'    LINE: UNKNOWN        EIBFN=X'0E0E'


ENTER:
PF1 : UNDEFINED           PF2 : UNDEFINED           PF3 : UNDEFINED
PF4 :                     PF5 :                     PF6 :
PF7 :                     PF8 :                     PF9 :
PF10:                     PF11: UNDEFINED           PF12:
```

Figure 56. EDF Example of the "Before" SQL EXEC Panel

```
TRANSACTION: XCO5 PROGRAM: TESTCO5 TASK:0000097 APPLID: CICS41 DISPLAY:00
 STATUS:  COMMAND EXECUTION COMPLETE
 CALL TO RESOURCE MANAGER DSNCSQL
 EXEC SQL OPEN                              P.AUTH=SYSADM  , S.AUTH=
 PLAN=TESTCO5,  DBRM=TESTCO5,  STMT=00221, SECT=00001
 SQL COMMUNICATION AREA:
  SQLCABC      = 136                                          AT X'03907C00'
  SQLCODE      =   -923                                       AT X'03907C04'
  SQLERRML     = 070                                          AT X'03907C08'
  SQLERRMC     = ' ACCESS,00000000,00000000,              '... AT X'03907C0A'
  SQLERRP      = 'DSNAET03'                                   AT X'03907C50'
  SQLERRD(1-6) =  000,  000, 00000, 0000000000, 00000,  000   AT X'03907C58'
  SQLWARN(0-A) = '_ _ _ _ _ _ _ _ _ _'                        AT X'03907C70'
  SQLSTATE     = 57015                                        AT X'03907C7B'


 OFFSET:X'001692'    LINE: UNKNOWN        EIBFN=X'0E0E'


ENTER:  CONTINUE
PF1 : UNDEFINED           PF2 : UNDEFINED           PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS   PF5 : WORKING STORAGE     PF6 : USER DISPLAY
PF7 : SCROLL BACK         PF8 : SCROLL FORWARD      PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY    PF11: UNDEFINED           PF12: ABEND USER TASK
```

Figure 57. EDF Example of the "After" SQL EXEC Panel

6.6.2 CICS Auxiliary Trace

The CICS auxiliary trace can be very effective in testing and debugging, because it provides trace entries for each SQL statement issued by the program. There is an additional trace entry where a nonzero return code is received for a SQL statement. The information provided in this additional entry can assist in problem determination. Refer to *IBM DATABASE 2 Diagnosis Guide and Reference* for a detailed description of the trace entries.

In CICS/ESA Version 4, the EXEC CICS ENTER TRACEID commands were replaced with CICS system trace. EXEC CICS ENTER TRACEID has a long pathlength, and the CICS system trace improves upon that.

Previously you had to have both the master trace flag on and the user trace flag on to get a CICS attach trace. With CICS/ESA Version 4, you have to set on only the file control trace flag. You can do this on the CICS/ESA Trace Control Facility panel shown in Figure 58 by pressing PF4 to select the Components option.

```
CETR             CICS/ESA Trace Control Facility       CICS CICS410

Type in your choices.

Item                                Choice       Possible choices

Internal Trace Status       ===>  STARTED      STArted, STOpped
Internal Trace Table Size   ===>  64      K    16K - 1048576K

Auxiliary Trace Status      ===>  STOPPED      STArted, STOpped, Paused
Auxiliary Trace Dataset     ===>  A            A, B
Auxiliary Switch Status     ===>  NEXT         NO, NExt, All

GTF Trace Status            ===>  STOPPED      STArted, STOpped

Master System Trace Flag    ===>  ON           ON, OFf
Master User Trace Flag      ===>  ON           ON, OFf

When finished, press ENTER.


PF1=Help      3=Quit      4=Components      5=Ter/Trn     9=Error List
```

Figure 58. Setting the File Control Flag with CETR

You then see the Component Trace Options panel shown in Figure 59 on page 116.

```
CETR            Component Trace Options                  CICS CICS410

Over-type where required and press ENTER.                PAGE 1 OF 3
Component Standard                          Special
--------  --------------------------------  -------------------------
  AP      1                                 1-2
  BF      1                                 1
  BM      1                                 1
  CP      1                                 1-2
  DC      1                                 1
  DD      1                                 1-2
  DI      1                                 1
  DM      1                                 1-2
  DS      1                                 1-2
  DU      1                                 1-2
  EI      1                                 1
  FC      1                                 1-2
  GC      1                                 1-2
  IC      1                                 1
  IS      1                                 1-2
  JC      1                                 1
  KC      1                                 1

PF:  1=Help   3=Quit   7=Back  8=Forward  9=Messages   ENTER=Change
```

Figure 59. CETR Component Trace Options Panel

FC indicates the file control trace flag and can be set to values of level 0,1, or 2. Zero turns off the CICS attachment facility trace, and 1 indicates that it should trace normally. Level 2 traces extra information.

The new CICS attachment facility trace contains more data, including the addresses of the DSN2LOT,DSN2SUB, and DSN2CCT control blocks. Exception traces and level 2 traces contain the entire DSN2LOT control block.

Within each trace entry, the parameter values are:

- Parm 1: An eye catcher
- Parm 2: History flags, return code
- Parm 3: The plan name
- Parm 4: DSN2LOT address
- Parm 5: DSN2CCT address—not available on dynamic plan traces
- Parm 6: DSN2SUB address—not available on dynamic plan traces
- Parm 7: The DSN2LOT block itself, up to the save area fields. You get Parm 7 only if you specify file control level 2, or if you receive an EXCEPTION trace.

6.6.3 CICS Transaction Dumps

Where transactions abend, you can use the dump to investigate the cause of the failure. The status of the SQL statements in the program can be found by analyzing working storage in the CICS EDF facility.

Refer to *IBM DATABASE 2 Diagnoses Guide and Reference* for details on interpreting dumps.

In CICS/ESA Version 4, a CICS-formatted region dump displays the information shown in Figure 60 on page 117 when a task is waiting to return from DB2. Figure 60 on page 117 shows some of the columns from the dispatcher domain section of a CICS/ESA 4.1-formatted region dump. The task with a KE_TASK of 02BA9080 shows a resource type of DB2 and has a resource name of CLOTECB. CLOTECB is the name of the ECB that the task is waiting on.

```
DS_TOKEN KE_TASK  TY S   P PS TT RESOURCE   RESOURCE ST   TIME OF
                                  TYPE       NAME          SUSPEND
00020003 029D7C80 SY SUS N OK -  TIEXPIRY   DS_NUDGE SUSP 18:50:29
000C0005 029CAC80 SY SUS N OK -                       SUSP 18:11:22
000E0005 02B99080 SY SUS N OK -  JCJOURDS   DFHJ01A  SUSP 18:39:39
00120007 02B99780 SY SUS N OK -  KCCOMPAT   SINGLE   OLDW 17:02:12
00160005 02B99B00 SY SUS N OK -  JCTERMN    SUBTASK  OLDW 17:01:56
00180003 029CA580 SY SUS N OK -  ICMIDNTE   DFHAPTIM SUSP 08:00:00
001A0003 029CA200 SY SUS N OK -  TCP_NORM   DFHZDSP  OLDW 18:51:27
001C0003 03F03900 SY SUS N OK -  ICEXPIRY   DFHAPTIX SUSP 18:50:29
008A0005 02B92080 SY SUS N OK -  JCJOURDS   DFHJ02A  SUSP 17:02:04
008E0001 02B13080 SY SUS N OK IN SMSYSTEM             SUSP 18:47:49
0102006F 02BA9080 NS SUS Y OK -  DB2        CLOTECB   MVS  18:51:22
01040031 02BA9780 NS RUN
01060009 02B13B00 NS SUS Y OK -                       MVS  18:50:29
```

Figure 60. CICS-Formatted Dump: Dispatcher Domain

You can see that CICS WAIT EXTERNAL in the DB2 Version 2 or Version 3 CICS attachment facility was replaced by the CICS XPI WAIT_MVS.

WAIT_MVS not only provides a shorter pathlength; it also helps you identify DB2 waits. Both CEMT INQUIRE TASK and CICS-formatted region dumps indicate a task that is waiting to return from DB2.

Figure 61 shows the output of a CEMT INQUIRE TASK command.

```
 INQUIRE TASK
 STATUS:  RESULTS - OVERTYPE TO MODIFY
  Tas(0000151) Tra(DSNC)           Sus Tas Pri( 255 )
? Tas(0000161) Tra(XCO5) Fac(1303) Sus Ter Pri( 001 )
  Tas(0000162) Tra(CEMT) Fac(1302) Run Ter Pri( 255 )
```

Figure 61. CICS CEMT INQUIRE Command for a CICS-DB2 Wait Transaction

When you place a question mark next to a specific task and then press enter, the CEMT output shows more detail about the task. Figure 62 on page 118 shows that task 161 is waiting to return from DB2.

```
INQUIRE TASK
SYNTAX OF SET COMMAND
 Tas(00000161) Tra(XC05) Fac(1303) Sus Ter Pri( 001 )
    Hty(DB2     ) Hva(CLOTECB ) Hti(000018) Sta(TO)
    Use(SYSADM  ) Rec(X'A8B5FE112D54CA85')
 CEMT  Set TAsk() | < All >
  < PRiority() >
  < PUrge | FOrcepurge >
```

Figure 62. CICS CEMT INQUIRE Command after Question Mark (?)

6.7 Going into Production

After designing, developing, and testing an application, you still need to perform other tasks before putting the application into production.

These tasks are highly dependent on the standards. For example, the tasks to be performed are different if:

- There are separate DB2 systems for test and production.
- Only one DB2 is used for both test and production.

The following discussion assumes that you use separate DB2 and CICS subsystems for test and production.

Going into production implies performing the following activities:

Use DDL to prepare production databases:

All DDL operations must run on the production DB2 system, using DDL statements from the test system as a base. Some modifications are probably needed, for example, increasing the primary and secondary allocations, as well as defining other volume serial numbers and defining a new VCAT in the CREATE STOGROUP statements.

Migrate DCLGEN:

For COBOL and PL/I programs, you may have to run DCLGEN operations on the production DB2 system, using DCLGEN input from the test DB2 system.

Depending on the option taken for compilations (if no compilations are run on the production system), an alternative could be to copy the DCLGEN output structures from the test libraries into the production libraries. This keeps all information separate between test and production systems.

Precompile: To produce DBRMs:

- Precompile CICS modules containing EXEC SQL statements on the production system, or
- Copy DBRMs from the test system to the production system libraries.

Compile and link-edit:
To produce load modules:

- Compile and link-edit the CICS modules on the production system, if DBRMs were produced by a precompilation, or
- Copy the load modules from the test system to the production system libraries, if the DBRMs were copied.

BIND: You must always perform the BIND process on the application programs on the production system.

GRANT EXECUTE:
You must grant users EXECUTE authority for the DB2 application plans on the production system.

Tests: Although no further tests should be necessary at this point, stress tests are useful and recommended to minimize the occurrence of resource contention, deadlocks, and timeouts and to check that the transaction response time is as expected.

CICS tables: To have new application programs ready to run, update the following RDO definitions on the CICS production system.

- RDO transaction definitions for new transaction codes
- RDO program definitions for new application programs and maps
- SIT for specific DB2 requirements, if it is the first DB2-oriented application going into production
- RCT fitting with the new application requirements
- SNT for IDs authorized to execute the new application plans.

In addition, if RACF is installed, you need to define new users and DB2 objects.

Additional considerations for going into production follow:

- *Type of threads*. When defining the new transactions and application plans in the RCT, you can use unprotected threads to get detailed accounting and performance information in the beginning. Later on, use protected threads as needed, and as recommended in Chapter 5, “Defining the Resource Control Table” on page 75.

- *BIND and time stamps*. An application program made of several modules usually becomes one application plan. Nevertheless, each module can be precompiled, compiled, and link-edited separately, all of them being bound at the same time.

  For each source module, the DB2 precompiler:

  – Creates a DBRM with a time stamp of Tdx (for example Td1 for the first source module, Td2 for the second source module, and so on)

  – Creates a modified source program with a time stamp of Tsx in the SQL parameter list (for example Ts1 and Ts2, if two source modules are involved).

  At BIND time, both modules in our example are bound to build the application plan. In addition, DB2 updates its catalog table SYSIBM.SYSDBRM with one line for each DBRM, altogether with time stamps Td1 and Td2.

  At execution time, DB2 checks the time stamps for each SQL statement. DB2 returns a -818 SQL code if:

  – Td1 and Ts1 are different, or

  – Td2 and Ts2 are different.

  To avoid -818 SQL codes when using multiple source programs bound in the same plan, you should:

  – Precompile, compile, and link-edit each module separately

  – Bind the complete plan, using all DBRMs. If you use a specific new or updated module in more than one application plan, you must bind all these application plans. Note that you must bind, and not rebind, the plans because some DBRMs changed.

- *Using PACKAGES*. There can be several methods to migrate from test to production. You can use whatever method is the most familiar and easy to use for you. Table 8 and Table 9 show one method of migration.

Table 8. Package Migration

Test System	Production System	Notes
	location_name.PROD_COLL.PRG3.VER1	The old version of the package
location_name.TEST_COLL.PRG3.VER2		A new version of the package is bound and then copied to the production system
	location_name.PROD_COLL.PRG3.VER1	The old version is still in the production collection
	location_name.PROD_COLL.PRG3.VER2	The new version is placed in the production collection

Table 9. Program Migration

Test System	Production System	Notes
	USER.PROD.LOADLIB(PGM3)	The original load module
USER.TEST.LOADLIB(PGM3)		The test load module
	USER.OLD.PROD.LOADLIB(PGM3)	The old version of the program is placed in other production library
	USER.PROD.LOADLIB(PGM3)	The new version of the program is placed in the production library

By selecting the production run library using the proper JCL, you can run either program. Then the correct version of the package is run, determined by the consistency token embedded in the program load module.

The previous example uses the VERSION keyword at precompile time. For a full explanation and use of the VERSION keyword, refer to *DB2 Packages: Implementation and Use*.

- *Using EXPLAIN*. Due to various factors, such as the sizes of tables and indexes, comparing the EXPLAIN output between test and production systems can be useless. Nevertheless, we recommend that you run EXPLAIN when you first bind the plan on the production system, to check the DB2 optimizer decisions.

Chapter 7. Monitoring, Tuning, and Handling Deadlocks

This chapter discusses monitoring, tuning, and deadlock handling in the CICS-DB2 environment. See also Chapter 10, “CICS-DB2 Application Design Considerations” on page 191.

7.1 Monitoring

The objective of monitoring the CICS attachment facility is to provide a basis for accounting and tuning. To achieve this objective, you need to obtain the following data:

- The number of transactions accessing DB2 resources
- The average number of SQL statements issued by a transaction
- The average processor usage for a transaction
- The average response time for a transaction
- The cost associated with particular transactions
- Buffer pool activity associated with a transaction
- Locking activity associated with a transaction. This includes whether table space locks are used instead of page locks, and whether lock escalation occurs, for example due to repeatable read.
- The level of thread usage for RCT entries
- The level of thread reuse for protected threads in the RCT.

You should also monitor your test environment to:

- Check that new programs function correctly (that is, use the correct call sequence) against test databases.
- Detect any performance problems due to excessive I/O operations or inefficient SQL statements.
- Detect bad design practices, such as holding DB2 resources across screen conversations.
- Set up optimum locking protocols to balance application isolation needs with those of existing applications.

Include monitoring in the acceptance procedures for new applications, so that any problems not detected during the test period can be quickly identified and corrected.

7.1.1 Monitoring Tools

You can use some, or all, of the following tools to monitor the CICS attachment facility and CICS transactions that access DB2 resources.

- Monitor the CICS attachment facility using:
  - CICS attachment facility commands
  - DB2 commands.
- Monitor CICS transactions using CICS auxiliary trace
- Monitor DB2 using:

  - DB2 statistics records
  - DB2 accounting records
  - DB2 performance records.
- Monitor the CICS system with the CICS monitoring facilities (CMF)
- Monitor the MVS system using the Resource Measurement Facility (RMF).

In the following sections, we provide a description of these facilities and the data they provide.

7.1.2 Monitoring the CICS Attachment Facility

You monitor the CICS attachment facility by using commands addressed to both DB2 and the CICS attachment facility itself.

7.1.2.1 Monitoring the CICS Attachment Facility Using Attachment Facility Commands

You can monitor and modify the status of the connection between CICS and DB2 using commands provided by the CICS attachment facility. The commands that you can enter are described later in this section and the examples of these commands use the RCT shown in Figure 63 on page 125.

- We define the DB2 subsystem name as DSN3, with the DB2 subsystem recognition character as %.
- We define THRDMAX as 20 in the TYPE=INIT statement. Note that the sum of all THRDS values must be less than or equal to the value of THRDMAX.
- We define a TYPE=COMD statement for DSNC. The authorization ID for DSNC commands is the 8-character USERID.
- The TYPE=POOL statement relates to the transaction IDs TST1 and TST2. We do not define a THRDS value.
- Each DSNCRCT TYPE=ENTRY uses:

  ```
  THRDM=5
  THRDA=3
  THRDS=3
  ```

  For performance and processor use, we recommend that you make THRDM, THRDA, and THRDS the same.
- Note that the DSNCRCT TYPE=ENTRY defined for plan ″DSN8CC13″ relates to transaction IDs C888 and D8CS. Note also that we do not define C888 in the RDO transaction definition.

```
         DSNCRCT TYPE=INIT,SUFFIX=3,SUBID=DSN3,                        X
               THRDMAX=20
*
         DSNCRCT TYPE=COMD,DPMODE=HIGH,TXID=DSNC,                      X
               AUTH=(USERID,*,*),                                      X
               ROLBE=NO,                                               X
               THRDM=1,                                                X
               THRDA=1,                                                X
               THRDS=1,                                                X
               TWAIT=POOL
*
         DSNCRCT TYPE=POOL,                                            X
               THRDM=5,THRDA=3,PLAN=DSN8CC13,TXID=(TST1,TST2)
*
*        DEFINE THE PLI SAMPLE TRANSACTION
*
         DSNCRCT TYPE=ENTRY,TXID=D8PS,THRDM=5,THRDA=3,PLAN=DSN8CP13,   X
               AUTH=(USERID,*,*),THRDS=3
         DSNCRCT TYPE=ENTRY,TXID=D8PP,THRDM=5,THRDA=3,PLAN=DSN8CQ13,   X
               AUTH=(USERID,*,*),THRDS=3
*
*        DEFINE THE COBOL SAMPLE TRANSACTION
*
         DSNCRCT TYPE=ENTRY,THRDM=5,THRDA=3,PLAN=DSN8CC13,             X
               AUTH=(USERID,*,*),THRDS=3,TXID=(C888,D8CS)
*
*        DEFINE THE PLI PHONE  TRANSACTION
*
         DSNCRCT TYPE=ENTRY,TXID=D8PT,THRDM=5,THRDA=3,PLAN=DSN8CH13,   X
               AUTH=(USERID,*,*),THRDS=3
*                D8PU TRANSACTION USED INTERNALLY
         DSNCRCT TYPE=ENTRY,TXID=D8PU,THRDM=5,THRDA=3,PLAN=DSN8CH13,   X
               AUTH=(USERID,*,*),THRDS=3
         DSNCRCT TYPE=FINAL
         END
```

Figure 63. Sample RCT Used for Monitoring the CICS Attachment Facility

The messages in the new CICS attachment facility are virtually the same as the messages in the old CICS attachment facility, except that the new CICS attachment facility messages start with DSN2. If you get a DSN2xxxx message, you can look it up under the DSNCxxxx message of the same number in *IBM DATABASE 2 Messages and Codes*. For new messages refer to section 3.4, “Handling Messages” on page 52.

DSNC DISC (Disconnect) Command

This command causes currently connected threads to be terminated as soon as they are not used by a transaction. It delays the termination of active threads until the CICS transaction ends, but it does not prevent the creation of new threads. This command:

- Frees DB2 resources shared by the CICS transactions.
- Allows exclusive access to these resources for utilities or data definition statements.

To guarantee that no new threads are created for a plan, all CICS-related transactions must be disabled before entering the DSNC DISC command. Note also that all transactions defined in the same RCT entry have the same plan.

Figure 64 on page 126 shows an example of this command and its output.

```
DSNC DISC DSN8CC13

DSNC021I THE DISCONNECT COMMAND IS COMPLETE
```

Figure 64. DSNC DISC Command Example. The disconnect command causes all transactions sharing the same plan to be disconnected. From an operational viewpoint, it can be an advantage to define only one transaction ID for each RCT entry.

DSNC DISP (Display) Command

Use this command to display information on the status of plans and transactions defined in the RCT, and statistics associated with each entry in the RCT.

The output from the display command is normally routed to the terminal that issued the command. The output may, however, be routed to other destinations.

Figure 65 on page 127 and Figure 66 on page 128 show examples of displaying the status of plans and transactions.

```
DSNC DISP PLAN


1. Two Threads still alive
================================================

 DSNC013I  DISPLAY REPORT FOLLOWS
   NAME        A/I     AUTH-ID

   DSN8CP13     I
   DSN8CC13     I

 DSNC020I  THE DISPLAY COMMAND IS COMPLETE


2. No more Threads alive
 ======================

 DSNC013I  DISPLAY REPORT FOLLOWS
   NAME        A/I     AUTH-ID

 DSNC041I NO ACTIVE THREADS
 DSNC020I  THE DISPLAY COMMAND IS COMPLETE
```

Figure 65. Sample Output for the DSNC DISP PLAN Command.
NAME is the four-character plan name.
A/I can be:
A, when the thread is within a unit of work
I, when the thread is waiting for a UR authorization ID for the plan being used

Figure 66 on page 128 shows that the DSNC DISP TRAN command provides similar information to the DSNC DISP PLAN command. The DSNC DISP PLAN command shows the plan name is shown on the report. The DSNC DISP TRAN command shows the first transaction ID associated with each plan.

```
DSNC DISP TRAN


1. Two Threads still alive
==========================================

 DSNC013I  DISPLAY REPORT FOLLOWS
   NAME        A/I     AUTH-ID

   D8PS         I
   C888         I

 DSNC020I  THE DISPLAY COMMAND IS COMPLETE


2. No more Threads alive
========================

 DSNC013I  DISPLAY REPORT FOLLOWS
   NAME        A/I     AUTH-ID

 DSNC041I NO ACTIVE THREADS
 DSNC020I  THE DISPLAY COMMAND IS COMPLETE
```

Figure 66. Sample Output for the DSNC DISP TRAN Command.
NAME is the 4-character transaction name.
A/I can be:
A, when the thread is within a unit of work.
I, when the thread is waiting for a UR authorization ID for the plan being used.

Note the second line showing C888 as the transaction ID. In our sample RCT (Figure 63 on page 125), you can see that C888 and D8CS share the same RCT entry. Even though C888 was not used and D8CS was used, only the first transaction ID for the TYPE=ENTRY statement is shown in the output.

Figure 67 on page 129 illustrates the statistics displayed for the DSNC DISP (Display) STAT (Statistics) command. These statistics are also sent to the destination specified in the RCT (default is CSMT), each time the CICS attachment facility is stopped. However, when using the DSNC DISP STAT command, only the first page is displayed at the issuing terminal. The total output is sent to the destination defined in the RCT (default: CSMT, usually defined as SYSOUT=* in the DD statement). The DSNC DISP STAT command does not provide any paging and scrolling capability if there is more than one page to display.

A possible circumvention could be:

1. Use DSNC MODI (modify) to change the current destination to a transient data queue.
2. Use the CEBR GET function to move the contents of the transient data queue into a temporary storage queue. CEBR can then be used to scroll backward and forward.

```
DSNC DISP STAT

DSN2014I   STATISTICS REPORT FOR 'DSN2CT03' FOLLOWS
                                                        -----COMMITS-----
TRAN  PLAN      CALLS   AUTHS    W/P HIGH   ABORTS   1-PHASE   2-PHASE
DSNC                5       1      0    1       0         0         0
TST1  DSN8CC13      0       0      0    0       0         0         0
D8PS  DSN8CP13    354       3      0    1       0        28         0
D8PP  DSN8CQ13    241       2      0    1       0        23         0
C888  DSN8CC13    391       3      0    1       0        31         0
D8PT  DSN8CH13    101       2      0    1       0         9         0
D8PU  DSN8CH13      2       2      0    1       0         2         0

DSN2020I THE DISPLAY COMMAND IS COMPLETE
```

Figure 67. Sample Output for CICS/ESA Version 4 Attachment Facility DISP STAT Command

The following statistics information is provided for each RCT entry (TYPE=COMD, TYPE=POOL, and each TYPE=ENTRY):

TRAN This column contains the first (or only) transaction ID of the RCT ENTRY. There is one row in the report for each RCT TYPE=ENTRY, TYPE=POOL, and TYPE=COMD.

PLAN This column contains the plan name associated with the RCT entry. If dynamic plan switching is enabled for this entry, the value is "********."

Note that the command processor transaction (DSNC) does ***not*** have a plan associated with it, because it uses a command thread.

CALLS This column contains the number of SQL calls issued by transactions associated with this RCT entry. This does not include EXEC CICS SYNCPOINT calls or EXEC CICS SYNCPOINT ROLLBACK calls.

You can determine the average number of SQL calls for the transactions using this entry from this value.

AUTHS This column contains the number of sign-ons executed by threads associated with this RCT entry. This does not indicate whether a new thread was created or an existing thread was reused.

A sign-on occurs when:

- A thread is first created
- The transaction ID changed
- The authorization ID changed
- The thread has an open CURSOR WITH HOLD
- The thread has modified special registers (CURRENT SQLID, CURRENT SERVER, and so on).
- TOKENE=YES or TOKENI=YES and the thread is reused.

The C888 entry in the figure shows that the thread was reused at least ten times by the same user-transaction pair. This is the reason for having 31 COMMITs (31 tasks) with only three AUTHs.

W/P This column contains the number of times all available threads for this RCT entry were busy and the transaction had to wait (TWAIT=YES) or was diverted to the pool (TWAIT=POOL).

This value, compared to the total number of transactions using this entry, shows if there are enough threads defined for this entry.

HIGH This column contains the maximum number of threads started for transactions associated with this RCT entry. This number includes transactions forced to wait or diverted to the pool. Note that threads can be started but never used. It provides a basis for setting the THRDM value for the entry.

ABORTS This column contains the number of EXEC CICS SYNCPOINT ROLLBACK calls associated with this RCT entry.

COMMITS Commit statistics contain a count of the number of EXEC CICS SYNCPOINT calls associated with each RCT entry. *1-PHASE* indicates the number of single-phase commits. This includes read-only commits (indicated by FRBRC2=F33100) and commits where DB2 is the only updater (indicated by CICS). For the latter case, a DSN3SYNC call is generated. *2-PHASE* indicates the number of prepare/commit sequences completed.

The number of commits performed equals the number of CICS tasks, provided that the program has only one syncpoint, which is the case in the figure. Transactions without DB2 calls are not counted here.

In this example, single-phase commit was used because DB2 is the only resource manager in the unit of recovery (UR). Refer to section 9.3, “Other Commit Protocols” on page 178 and section 9.2, “Two-Phase Commit Protocol” on page 172 for details in commit protocols.

Notes:

1. There is no information in these statistics on the level of *thread reuse*.
2. This data provides the necessary information to modify the CICS attachment facility characteristics to meet installation objectives.

DSNC MODI (Modify) Command

This command allows you to change the characteristics of the connection to DB2 by modifying parameters in the RCT. You can change the RCT-defined CICS destination where the CICS attachment facility error messages are sent. You can also use DSNC MODI to increase or decrease the value of THRDA as defined in the RCT for a given transaction ID. The upper limit for this change is the THRDM value for this RCT group or entry.

We do not recommend increasing or decreasing the value of THRDA, from performance and processor consumption viewpoints.

Figure 68 on page 131 shows an example of this command and its output. Refer to the *IBM DATABASE 2 Command and Utility Reference*, for full details on these commands.

The use of these CICS attachment facility commands should be controlled through the standard CICS security mechanism. No authorization checking by DB2 is involved.

```
DSNC MODI DEST CSMT CSSL

  DSNC019I THE MODIFY COMMAND IS COMPLETE


DSNC MODI TRAN D8CS 3

  DSNC019I THE MODIFY COMMAND IS COMPLETE
```

Figure 68. DSNC MODI Command Examples

Note on Figure 68: If you do not know the destination being used for the CICS attachment facility error messages, use the command:

```
dsnc modi dest dddd csmt
```

Do not define the 'DDDD' destination in the DCT or in the RCT. You then get the messages:

```
DSNC006I THE 'DDDD' DESTINATION ID IS INVALID
DSNC039I THE ERROR DESTINATION ARE 'CSSL **** ****'
```

'CSSL' is the currently used destination-ID.

7.1.2.2 Monitoring the CICS Attachment Facility Using DB2 Commands

Once a connection between CICS and DB2 is established, terminal users authorized by CICS security can use the DSNC transaction to route commands to the DB2 system. These commands are of the form:

```
DSNC -DB2command
```

The command is routed to DB2 for processing. DB2 checks that the authorization ID passed from CICS is authorized to issue the command entered.

Responses are routed back to the originating CICS user. The command recognition character (CRC) **-** must be used to distinguish DB2 commands from CICS attachment facility commands. For DB2 commands issued from CICS, the CRC is always **-**, regardless of the subsystem recognition character.

Both CICS and DB2 authorization are required to issue DB2 commands from a CICS terminal:

- CICS authorization is required to use the DSNC transaction, and
- DB2 authorization is required to issue DB2 commands.

The output from a DSNC -DISPLAY THREAD(SCMCICSA) command is shown in Figure 69 on page 132 and discussed below. For performance reasons we recommend that you always qualify your DISPLAY THREAD command rather

than using DSNC -DISPLAY THREAD(*). In this case SCMCICSA is the name of our CICS region.

```
DSNC -DISPLAY THREAD(SCMCICSA)

  --------------------
DSNV401I % DISPLAY THREAD REPORT FOLLOWS -
DSNV402I % ACTIVE THREADS -
NAME     ST  A  REQ  ID          AUTHID    PLAN     ASID
SCMCICSA  N       3             R7322CI             0013
SCMCICSA  N     296  GTOOD8PS   R7322JE             0013
SCMCICSA  N       0                                 0013
SCMCICSA  N       0                                 0013
SCMCICSA  N     301  GTOOD8PP   R7322JE             0013
SCMCICSA  N       0                                 0013
SCMCICSA  N       0                                 0013
SCMCICSA  T     343  GTOOD8CS   R7322JE  DSN8CC13   0013
SCMCICSA  N       0                                 0013
SCMCICSA  N       0                                 0013
SCMCICSA  T     132  GTOOD8PT   R7322JE  DSN8CH13   0013
SCMCICSA  N       0                                 0013
SCMCICSA  N       0                                 0013
SCMCICSA  N      12  GTOOD8PU   R7322JE             0013
SCMCICSA  N       0                                 0013
SCMCICSA  N       0                                 0013
SCMCICSA  T  *   15  GCOODSNC   R7322CI             0013
DSN9022I % DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION
```

Figure 69. Sample Output for the DB2 DISPLAY THREAD Command

NAME A 1- to 8-character variable representing the connection name. With CICS, it is the CICS VTAM application name, as defined in the SIT APPLID parameter.

STatus A 1- or 2-character connection status code, as follows:

- N The thread is in either IDENTIFY or SIGNON status.
- QT The CREATE THREAD request has been queued.
- T A thread has been established (the plan allocated).
- QD The thread is queued for termination.
- D The thread is in the process of termination.

If you have active threads with QD or D status codes, it could be a DB2 problem.

A The * if the thread is in use by a transaction and active in DB2.

REQ A wraparound counter to show the number of DB2 requests.

ID A 1- to 8-character variable representing the recovery correlation ID associated with the thread. See Figure 89 on page 186 for the layout of the correlation ID.

AUTHID The authorization ID associated with a sign-on connection. If the connection is not signed on, this field is blank.

PLAN A 1- to 8-character variable representing the plan name associated with the thread. If a thread was not established, this field is blank. For a command, as in our example, there is no plan displayed by the DSNC -DISPLAY THREAD(*) command.

ASID A 1- to 4-character hexadecimal number representing the address space identifier (ASID) of the CICS address space. The descriptions of DB2 messages DSNV401 and DSNV402 give more information about the ASID.

Notes on Figure 69 on page 132:

- The threads displayed are any TCB that is identified to DB2.
- The % character shown in messages DSNV401I and DSNV402I is the subsystem recognition character of the DB2 subsystem CICS is connected to.
- The sample output shows 17 lines of information. They correspond to the protected threads as defined in the THRDS parameter of the RCT, plus:
  - One used at CICS attachment facility startup time
  - One for the command thread.
- Two plans are still allocated (T in ST field, plus a plan name).
- Only one thread is active (* in the A column).
- Two entries still have a plan name. It means that a thread still exists and can be reused.
- When a thread is reused by a new authorization ID, only this new authorization ID appears in the AUTHID field.
- Using successive DSNC -DISPLAY THREAD commands allows you to see if the request counter is being incremented.
- The entry (last line of output before the DSN9022I message) for CICS is for the command thread being used to process the display command. The details are:
  - The name displayed. In this case the connection-name is SCMCICSA, as defined as APPLID parameter in the CICS SIT.
  - The ID refers to the correlation ID, which consists of the thread number concatenated to the transaction ID. In this case, the thread number is GC00 (C for the command thread type), and the transaction ID is that of the CICS attachment facility command processor (DSNC). See Figure 89 on page 186 for details.
  - There is no plan displayed for the DSNC transaction.

7.1.3 Monitoring the CICS Transactions

You can use the CICS auxiliary trace facility to trace SQL calls issued by a CICS application program.

Whenever control returns to the CICS attachment facility from DB2, an entry is made in the CICS trace table. The trace ID for this entry is specified in the RCT (default is X′C0′, X′C1′, X′C2′ or 192, 193, 194 decimal for releases previous to CICS/ESA Version 4 and X′1C0′, X′1C1′, X′1C2′ or 448, 449, 450 decimal for CICS/ESA Version 4).

These entries are shown in the trace between a call to the CICS TRUE (ERM PASSING CONTROL TO RM (AP00E7)) and a return from the TRUE (ERM REGAINING CONTROL FROM RM (AP00E7)). Note that CICS uses ERM (external resource manager) in the trace listing referring to TRUEs. In this case, RM (resource manager) refers to DB2. Any trace entries between these two reflect requests for CICS resources made by the CICS attachment facility to pass the request to DB2.

```
AP 00E7 ERM ENTRY APPLICATION                       REQ(0104) FIELD-A( AA448F5B ...$) FIELD-B(22C43282 .D.b) RESOURCE(DSNCSQL )

             TASK-00054 KE_NUM-0016 TCB-006D8750 RET-83C016B4 TIME-10:5 0:35.5081335007 INTERVAL-00.0000860312    =000468=

XS 0301 XSIS  ENTRY - FUNCTION(INQ_SECURITY_DOMAIN_PARMS)

             TASK-00054 KE_NUM-0016 TCB-006D8750 RET-8009B4F0 TIME-10:50:35.5082500319 INTERVAL-00.0001165312    =000469=

               1-0000  00D00000 000000B4 00000000 00000000  B0200000 00000000 03000000 00000000  *.............................*
                 0020  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *.............................*
                 0040  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *.............................*
                 0060  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *.............................*
                 0080  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *.............................*
                 00A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *.............................*
                 00C0  00000000 00000000 00000000 00000000                                      *.............               *

XS 0302 XSIS  EXIT - FUNCTION(INQ_SECURITY_DOMAIN_PARMS) RESPONSE(OK) SECURITY(YES)

             TASK-00054 KE_NUM-0016 TCB-006D8750 RET-8009B4F0 TIME-10:50:35.5082553132 INTERVAL-00.0000052812    =000470=
               1-0000  00D00000 000000B4 00000000 00000000  B0200000 00000000 03000100 00000000  *.............................*
                 0020  00000000 00000000 00000000 C3C9C3E2  F4F14040 40404040 40404040 0102020 0  *............CICS41          ....*
                 0040  01010201 00000000 00000000 00000000  00000000 00000000 00000000 D1C3C9C3  *.........................JCIC*
                 0060  E2D1C3E3 00000000 00000000 00000000  00000000 D7C3C9C3 E2D7E2C2 00000000  *SJCT..............PCICSPSB....*
                 0080  00000000 E2C3C9C3 E2E3E2E3 00000000  00000000 00000000 00000000 00000000  *....SCICSTST.................*
                 00A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *.............................*
                 00C0  00000000 00000000 00000000 00000000                                      *.............               *

AP 00E7 ERM EVENT PASSING-CONTROL-TO-RM              REQ(4004) FIELD-A(AA448F5B ...$) FIELD-B(22C43282 .D.b) RESOURCE(DSNCSQL )

             TASK-00054 KE_NUM-0016 TCB-006D8750 RET-83C016B4 TIME-10:50:35.5082825007 INTERVAL-00.0000271875    =000471=

AP 00E1 EIP ENTRY ASSIGN                            REQ(0004) FIELD-A(03B1E21C ..S.) FIELD-B(08000208 ....)

             TASK-00054 KE_NUM-0016 TCB-006D8750 RET-838B6A9C TIME-10:50:35.5083667507 INTERVAL-00.0000842500    =000472=

US 0401 USXM  ENTRY - FUNCTION(INQUIRE_TRANSACTION_USER)

             TASK-00054 KE_NUM-0016 TCB-006D8750 RET-837E454C TIME-10:50:35.5083793132 INTERVAL-00.0000125625    =000473=
               1-0000  00B80000 000000C0 00000000 00000000  B01B5580 00000000 06000000 00000000  *.............................*
                 0020  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *.............................*
                 0040  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *.............................*
                 0060  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *.............................*
                 0080  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *.............................*
                 00A0  00000000 00000000 00000000 00000000  00000000 00000000                    *.......................       *

US 0402 USXM  EXIT - FUNCTION(INQUIRE_TRANSACTION_USER) RESPONSE(OK) USERID_LENGTH(6) USERID(SYSADM) CURRENT_GROUPID_LENGTH(4)
                   CURRENT_GROUPID(SYS1) NATIONAL_LANGUAGE() USERNAME(####################) OPERATOR_IDENT() OPERATOR_PRIORITY(0)
                   OPERATOR_CLASSES(000001)

             TASK-00054 KE_NUM-0016 TCB-006D8750 RET-837E454C TIME-10:50:35.5087372819 INTERVAL-00.0003579687    =000474=
               1-0000  00B80000 000000C0 00000000 00000000  B01B5580 00000000 06000100 00000003  *.............................*
                 0020  00000000 00000000 00000000 00000000  00000000 06E2E8E2 C1C4D440 40404000  *.....................SYSADM    .*
                 0040  04E2E8E2 F1404040 40404000 40404000  7B7B7B7B 7B7B7B7B 7B7B7B7B 7B7B7B7B  *.SYS1      .   .################*
                 0060  7B7B7B7B C3C9C3E2 F4F14040 40404001  00000001 00000000 7F6A0438 00000000  *####CICS41     ........”.......*
                 0080  00000000 00000000 00000000 00000000  00000000 00000000 00000000 E3F1F3F0  *............................T130*
                 00A0  F3404040 01000000 00000000 00000000  00000000 00000000                    *3   .....................       *

AP 00E1 EIP EXIT ASSIGN OK                          REQ(00F4) FIELD-A(00000000 ....) FIELD-B(00000208 ....)

             TASK-00054 KE_NUM-0016 TCB-006D8750 RET-838B6A9C TIME-10:50:35.5087452819 INTERVAL-00.0000080000    =000475=
```

Figure 70 (Part 1 of 2). CICS/ESA Version 4 Auxiliary Trace Entry for an SQL Statement

Figure 70 on page 134 shows the format of the trace entries for an SQL application. The format of these trace entries is described in the *IBM DATABASE 2 Diagnosis Reference Volume 4: Service Traces* for DB2 2.3, or *IBM DATABASE 2 Diagnosis Guide and Reference* for DB2 3.1.

Traces for CICS/ESA Version 3 are quite similar to the ones shown in Figure 70. However, the information of DSN2LOT is not there for exception reports, or when trace LEVEL 2 is requested.

7.1.4 Monitoring DB2 when Used with CICS

Using SMF and/or GTF records produced by DB2, the user can monitor DB2 when used with CICS. The DB2 Performance Monitor program product is useful to provide reports based on:

- Statistics records
- Accounting records
- Performance records.

```
DS 0004 DSSR  ENTRY - FUNCTION(WAIT_MVS) RESOURCE_NAME(CLOTECB) RESOURCE_TYPE(DB2) ECB_ADDRESS(03B1E0F8) PURGEABLE(YES)
                   WLM_WAIT_TYPE(OTHER_PRODUCT)

              TASK-00054 KE_NUM-0016 TCB-006D8750 RET-838B704E TIME-10:50:35.5087750632 INTERVAL-00.0000297812    =000476=
                1-0000  00580000 00000014 00000001 00000000  B3242000 00000000 06000000 00000000  *.............................*
                  0020  00000000 00000000 C3D3D6E3 C5C3C240  C4C2F240 40404040 00000000 00000000  *........CLOTECB DB2     ........*
                  0040  03B1E0F8 00000000 00010000 00000A00  00000000 00000000                    *...8....................       *

DS 0005 DSSR  EXIT - FUNCTION(WAIT_MVS) RESPONSE(OK)

              TASK-00054 KE_NUM-0016 TCB-006D8750 RET-838B704E TIME-10:50:35.5228140319 INTERVAL-00.0140389687*   =000477=
                1-0000  00580000 00000014 00000001 00000000  B3242000 00000000 06000100 00000000  *.............................*
                  0020  00000000 00000000 C3D3D6E3 C5C3C240  C4C2F240 40404040 00000000 00000000  *........CLOTECB DB2     ........*
                  0040  03B1E0F8 00000000 00010000 00000A00  00000000 00000000                    *...8....................       *

AP 01C0 USER  EVENT - USER-EXIT-PROGRAM-ENTRY

              TASK-00054 KE_NUM-0016 TCB-006D8750 RET-83C016B4 TIME-10:50:35.5228240944 INTERVAL-00.0000100625    =000478=
                1-0000  C4C2F240 60405CC5 E7C35C                                        *DB2 - *EXC*                    *
                2-0000  81000004 00F30040                                              *a....3.                        *
                3-0000  E3C5E2E3 C3F0F540                                              *TESTC05                        *
                4-0000  03B1E09C                                                       *....                           *
                5-0000  00000000                                                       *....                           *
                6-0000  04A07A80                                                       *..:.                           *
                7-0000  C4E2D5F2 D3D6E340 000B0000 000116B8  00000000 04A07A80 E2E8E2C1 C4D44040  *DSN2LOT ..............:.SYSADM  *
                  0020  40404040 40404040 E7C3F0F5 00000000  00000000 00000000 00000000 C4C2F2D5  *        XC05.............DB2N*
                  0040  C5E34040 E3F1F3F0 F3404040 448F5B22  C5500000 81000004 80000420 00000000  *ET  T1303   ..$.E&..a..........*
                  0060  00000000 00F30040 81000004 00F30040  03A003D0 00000000 00000000 00000000  *.....3. a....3. ................*
                  0080  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                  00A0  E3C5E2E3 C3F0F540 01C00000 00000000  00000000 00000000 00000000 00000000  *TESTC05 ........................*
                  00C0  00000000 0000                                                    *......                         *

AP 01C0 USER  EVENT - USER-EXIT-PROGRAM-ENTRY

              TASK-00054 KE_NUM-0016 TCB-006D8750 RET-83C016B4 TIME-10:50:35.5228305944 INTERVAL-00.0000065000    =000479=
                1-0000  C4C2F240 60405CC5 E7C35C                                        *DB2 - *EXC*                    *
                2-0000  E7C3F0F5 80008020                                              *XC05....                       *
                3-0000  E3C5E2E3 C3F0F540                                              *TESTC05                        *
                4-0000  03B1E09C                                                       *....                           *
                5-0000  00000000                                                       *....                           *
                6-0000  04A07A80                                                       *..:.                           *
                7-0000  C4E2D5F2 D3D6E340 000B0000 000116B8  00000000 04A07A80 E2E8E2C1 C4D44040  *DSN2LOT ..............:.SYSADM  *
                  0020  40404040 40404040 E7C3F0F5 00000000  00000000 00000000 00000000 C4C2F2D5  *        XC05.............DB2N*
                  0040  C5E34040 E3F1F3F0 F3404040 448F5B22  C5500000 00810004 80008020 00000000  *ET  T1303   ..$.E&...a.........*
                  0060  00000000 00F30040 E7C3F0F5 80008020  03A003D0 00000000 00000000 00000000  *.....3. XC05....................*
                  0080  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                  00A0  E3C5E2E3 C3F0F540 01C00000 00000000  00000000 00000000 00000000 00000000  *TESTC05 ........................*
                  00C0  00000000 0000                                                    *......                         *


AP 00E7 ERM EVENT REGAINING-CONTROL-FROM-RM          REQ(4104) FIELD-A(AA448F5B ...$) FIELD-B(22C43282 .D.b) RESOURCE(DSNCSQL )

              TASK-00054 KE_NUM-0016 TCB-006D8750 RET-83C016B4 TIME-10:50:38.0113530319 INTERVAL-00.0000085000    =002327=

AP 00E7 ERM EXIT  APPLICATION                        REQ(1104) FIELD-A(AA448F5B ...$) FIELD-B(22C43282 .D.b) RESOURCE(DSNCSQL )

              TASK-00054 KE_NUM-0016 TCB-006D8750 RET-83C016B4 TIME-10:50:38.0113650319 INTERVAL-00.0000120000    =002328=
```

Figure 70 (Part 2 of 2). CICS/ESA Version 4 Auxiliary Trace Entry for an SQL Statement

The following reports are shown as examples. Refer to the documentation of the DB2PM release you are using for the format and meaning of the fields involved in the reports.

7.1.4.1 Using the DB2 Statistics Facility

DB2 produces statistical data on a subsystem basis at the end of each time interval, as specified at installation time. This data is collected and written to the SMF and GTF data set only if the facility is active. Refer to section 3.8, “Starting System Monitoring Facility for Accounting, Statistics, and Performance” on page 58 and to section 3.10, “Starting Generalized Trace Facility for Accounting, Statistics, and Performance” on page 59 for details on how to activate these facilities and direct the output to SMF and GTF.

Data related to the system services address space is written as SMF instrumentation facility component identifier (IFCID) 0001 records. Data related to the database services address space is written as SMF IFCID 0002 records. Refer to *IBM DATABASE 2 Administration Guide* for a description of these records.

These statistics are useful for tuning the DB2 subsystem, since they reflect the activity for all subsystems connected to DB2.

It is difficult to interpret this data when more than one subsystem (that is, CICS and TSO) is connected to DB2. However, the counts obtained while running the CICS attachment facility in a controlled environment (that is with CICS as the only subsystem connected, or with limited TSO activity) can be very useful.

The *IBM DATABASE 2 Administration Guide* shows and analyzes, from a DB2 viewpoint, the statistical data reported for the database and system services address spaces. Included here is a reduced version of the statistics summary report. You can use this report to monitor the average CICS transaction. Figure 71 on page 137 shows a sample of the report provided by DB2PM. Refer to the *IBM DATABASE 2 Administration Guide* for additional information on these reports.

Figure 71 on page 137 provides the following information:

- *SQL Manipulative* can be used to monitor the average transaction. In the example, the number of transactions about equals the number of threads, because the number of occurrences per thread and per commit is almost the same. It also means that there is no thread reuse. Otherwise, this number must be derived from the CICS statistics. In our case, the average transaction uses 4.4 SELECTs, 0.1 INSERTs, 0.6 UPDATEs, 0.1 DELETEs, 0.7 OPENs/CLOSEs of a cursor, and 4.3 FETCHs
- *SQL Control* can be used to check whether the application is using LOCK statements.
- *Buffer Pool* provides valuable information on:
  - Whether buffer expansions are occurring (Buffer Pool Expansions).
  - The number of data sets opened (Datasets Opened)
  - The number of pages retrieved (Getpage Requests)
  - Number of I/Os (Read Operations and Write I/O Operations)
- Reuse of threads can be estimated by comparing thread allocations (the Allocation Successful line under Service Controller) to the number of CICS transactions using the attach (from CICS statistics). Refer to section 7.1.4.6, “Monitoring Thread Reuse” on page 145.

  For performance, we recommend thread reuse in CICS environments, to avoid the overhead of creating a thread for each CICS transaction.
- Under *Locking*, monitor the number of timeouts and deadlocks. For deadlocks, more information is provided in section 7.3, “Handling Deadlocks” on page 148.

This statistical data is not checkpointed and is not retained by DB2 across restarts.

7.1.4.2 Using the DB2 Accounting Facility

The accounting facility provides detailed statistics on the use of DB2 resources by CICS transactions. This record is the basis for all accounting and tuning of DB2 resources used by CICS transactions.

DB2 gathers accounting data on an authorization ID within thread basis. When requested, the accounting facility collects this data and directs it to SMF, GTF, or both when the thread is terminated or when the authorization ID is changed.

```
                    DB2 PERFORMANCE MONITOR (V1 R0) DB2 ID=DSN3
                                  STATISTICS SUMMARY
CLASS ACTIVE TIME       12.43.07
ELAPSED TIME            10.38.07  THREADS  24  THREADS/MINUTE  2.1
S Q L MANIPULATIVE          # OCCUR.       /MINUTE        /THREAD       /COMMIT
----------------------------------     ---------      ---------     ---------
SELECT                            98          8.9            4.0           4.4
INSERT                             4          0.3            0.1           0.1
UPDATE                            15          1.3            0.6           0.6
DELETE                             4          0.3            0.1           0.1
PREPARE                            0          0.0            0.0           0.0
                  SUB-TOTAL      121         11.0            5.0           5.5
DESCRIBE                           0          0.0            0.0           0.0
OPEN CURSOR                       17          1.5            0.7           0.7
CLOSE CURSOR                      17          1.5            0.7           0.7
FETCH                             96          8.7            4.0           4.3
                  SUB-TOTAL      130         11.8            5.4           5.9
                      TOTAL      251         22.8           10.4          11.4
S Q L CONTROL               # OCCUR.       /MINUTE        /THREAD       /COMMIT
----------------------------------     ---------      ---------     --------
LOCK TABLE                         0          0.0            0.0           0.0
GRANT                              0          0.0            0.0           0.0
REVOKE                             0          0.0            0.0           0.0
INCREMENTAL BIND                   0          0.0            0.0           0.0
COMMENT ON                         0          0.0            0.0           0.0
LABEL ON                           0          0.0            0.0           0.0
                  TOTAL            0          0.0            0.0           0.0
                         STATISTICS SUMMARY
BUFFER POOL 00              # OCCUR.       /MINUTE        /THREAD       /COMMIT
----------------------------------     ---------      -------       --------
BUFFER POOL SIZE (KBYTES)  800
CURRENT ACTIVE BUFFERS      11
MAXIMUM NUMBER OF BUFFERS  200
MINIMUM NUMBER OF BUFFERS  200
BUFFER POOL EXPANSIONS             0          0.0            0.0           0.0
EXPANDED TO LIMIT                  0          0.0            0.0           0.0
STORAGE UNAVAILABLE                0          0.0            0.0           0.0
DATASETS OPENED                   20          1.8            0.8           0.9
READ OPERATIONS
----------------------------------
GETPAGE REQUESTS     (GET)      1738        158.0           72.4          79.0
READ I/O OPERATIONS  (RIO)        72          6.5            3.0           3.2
READS WITH PAGING                  0          0.0            0.0           0.0
PREFETCH REQUESTED                72          6.5            3.0           3.2
PAGE READ DUE TO SEQ PREFETCH     23          2.0            0.9           1.0
PREFETCH DISABLED-NO BUFFER        0          0.0            0.0           0.0
PREFETCH DISABLED-NO READ ENGINE   0          0.0            0.0           0.0
WRITE OPERATIONS
----------------------------------
SYSTEM PAGE UPDATES  (SWS)       361         32.8           15.0          16.4
SYSTEM PAGES WRITTEN (PWS)         0          0.0            0.0           0.0
WRITE I/O OPERATIONS (WIO)         0          0.0            0.0           0.0
WRITES WITH PAGING                 0          0.0            0.0           0.0
WRITE ENGINE NOT AVAIL FOR I/O     0          0.0            0.0           0.0
DEFERRED WRITE THRESHOLD REACHED   0          0.0            0.0           0.0
DM CRITICAL THRESHOLD REACHED      0          0.0            0.0           0.0
IMMEDIATE WRITES                   0          0.0            0.0           0.0
```

Figure 71 (Part 1 of 2). Sample Statistics Summary Report

The procedures involved in collecting this data are described in section 3.8, “Starting System Monitoring Facility for Accounting, Statistics, and Performance” on page 58 and in section 3.10, “Starting Generalized Trace Facility for Accounting, Statistics, and Performance” on page 59.

Refer to the *IBM DATABASE 2 Administration Guide*, for details on the general structure of DB2 SMF and GTF records, and descriptions of the fields in the accounting record.

```
EDM POOL                    # OCCUR.      /MINUTE        /THREAD       /COMMIT
-----------------------------------      --------       --------      --------
PAGES IN EDM POOL                412
PAGES USED FOR CT                  0
FREE PG IN FREE CHAIN            382
PAGES USED FOR DBD                 9
PAGES USED FOR SKCT               21
FAILS DUE TO POOL FULL             0          0.0            0.0           0.0
REQ FOR CT SECTIONS              160         14.5            6.6           7.2
LOAD CT SECT FROM DASD            42          3.8            1.7           1.9
REQUESTS FOR DBD                  95          8.6            3.9           4.3
LOADING DBD FROM DASD              2          0.1            0.0           0.0
SERVICE CONTROLLER          # OCCUR.      /MINUTE        /THREAD       /COMMIT
-----------------------------------      --------       --------      --------
MAXIMUM DB2 DATASETS EXPECTED      0
DATASETS OPEN - CURRENT           20
DATASETS OPEN - MAXIMUM           15
ALLOCATION ATTEMPTS               22          2.0            0.9           1.0
ALLOCATION SUCCESSFUL             22          2.0            0.9           1.0
AUTHORIZATION ATTEMPTS            32          2.9            1.3           1.4
AUTHORIZATION SUCCESSFUL          32          2.9            1.3           1.4
PLANS BOUND                        0          0.0            0.0           0.0
BIND ADD SUB-COMMANDS              0          0.0            0.0           0.0
BIND REPLACE SUB-COMMANDS          0          0.0            0.0           0.0
TEST BINDS NO PLAN-ID              0          0.0            0.0           0.0
AUTOMATIC BIND ATTEMPTS            0          0.0            0.0           0.0
AUTOMATIC BINDS SUCCESSFUL         0          0.0            0.0           0.0
AUTOMATIC BIND INVAL RESOURCE IDS  0          0.0            0.0           0.0
REBIND SUB-COMMANDS                0          0.0            0.0           0.0
ATTEMPTS TO REBIND A PLAN          0          0.0            0.0           0.0
PLANS REBOUND                      0          0.0            0.0           0.0
FREE SUB-COMMANDS                  0          0.0            0.0           0.0
ATTEMPTS TO FREE A PLAN            0          0.0            0.0           0.0
PLANS FREED                        0          0.0            0.0           0.0


                                        STATISTICS SUMMARY
CLASS ACTIVE TIME       12.43.07
ELAPSED TIME            10.38.07  THREADS    24  THREADS/MINUTE  2.1
                           STATS          TRACE       TRACE AET
SYSTEM EVENTS               # OCCUR.       # OCCUR.        SS.TH
------------------          --------       --------        ------
COMMANDS                          10              4          0.02
CHECKPOINTS                        0              0
EDM FULL                           0              0
SHORT ON STORAGE                   0              0
ABENDS EOM                         0              0
ABENDS EOT                         0              0
                              TCB TIME    SRB TIME TOTAL TIME     /THREAD     /COMMIT
CPU TIMES                        SS.TH       SS.TH      SS.TH       SS.TH       SS.TH
----------------------------     -------     ------     ------      ------      ------
SYSTEM SERVICES ADDRESS SPACE       0.11       0.19       0.30        0.01        0.01
DATABASE SERVICES ADDRESS SPACE     0.28       0.22       0.50        0.02        0.02
IRLM                                0.00       0.01       0.01        0.00        0.00
                           TOTAL    0.39       0.42       0.81        0.03        0.04
```

Figure 71 (Part 2 of 2). Sample Statistics Summary Report

The accounting facility output for a single transaction can be used for monitoring and tuning purposes. Using the accounting facility for accounting purposes depends mainly on the installation requirements. Section 8.4, “Accounting Considerations for DB2” on page 161 gives considerations on how to use this function to account for the use of DB2 resources to users.

The identification section of the accounting record written to SMF and GTF provides a number of keys on which the data can be sorted and summarized. These include the authorization ID, the transaction ID and the plan name.

Summarizing by authorization ID makes it easier to correlate the output from the accounting facility with that of the CICS monitoring facilities. This correlation is required to give an overall usage of resources by CICS transactions in the CICS and DB2 address spaces.

Figure 72 on page 140 is a listing of the accounting output produced for a sample CICS transaction.

As shown in the figure, the report provides:

- A header line that identifies the subsystem.
- Below the class active time, we can see that the report has been produced by authorization ID.
- *Application time (Class 1)*. This section contains information on the elapsed time covered by this report, and the processor (CPU) time used during this elapsed time. The processor time is split into TCB and SRB times. See Chapter 8, “Accounting” on page 153 to interpret these figures.
- In the *Highlights* section, the first line (#OCCUR**.**) shows the number of accounting records that were written.
- *Buffer manager summary*. This section contains information on the number of page references in the four buffer pools. This includes the number of get page requests, buffer pool expansions, and system page updates. Note that each page reference recorded may cause an I/O. Synchronous read I/Os are recorded. Write I/Os are not recorded.
- *Locking summary*. This section contains information related to the use of locks. The number of lock suspensions, deadlocks, and timeouts are recorded.
- *Manipulative SQL activity*. This section contains information on the number of times each type of SQL statement was executed. For example, in our case, fifteen updates were done. These counts provide a method for checking that application programs conform to their specifications and installation standards.
- *Control SQL activity*. This section contains information about control SQL activity. There is usually no control SQL activity in a CICS-DB2 system.
- *Definitional SQL activity*. This section contains information about definitional SQL activity. There is usually no definitional SQL activity in a CICS-DB2 system.

7.1.4.3 Package Accounting and Performance

Accounting information for packages is not available in DB2 2.3 but is in DB2 3.1. Along with the DB2 3.1 enhancement for package accounting support, performance analysis tools are enhanced to support packages as well.

7.1.4.4 DB2PM Version 3

DB2PM version 3 along with DB2 3.1 have package accounting support. Two DB2PM identifiers, MAINPACK and PACKAGE, were introduced for this purpose:

- MAINPACK can be used to distinguish plans according to the packages they contain. The representative package is either the first or last package or DBRM in a plan. This identifier is useful when the name of a plan does not provide satisfactory identification, for example, reporting database authority

```
                    DB2 PERFORMANCE MONITOR   DB2 ID=DSN3

                  ACCOUNTING DETAIL REQUESTED FROM NOT SPECIFIED
                                              TO   NOT SPECIFIED
ACTUAL FROM    23:41:00.00
       TO      23:54:00.00

CLASS ACTIVE TIME 13:00.00
                     BY AUTHID
AUTHID - R7322CI     APPLICATION
                        TIME       DB2 TIMES                              TOTALS
AVERAGE                (CLASS 1)    (CLASS 2)     HIGHLIGHTS:             OR MEANS
--------------------------------   -----------   ----------------------  --------
ELAPSED TIME / INVOC     1.05.16     1.00.93    #OCCUR.                       23
CPU TIME (SSSSSS.TH)        0.09        0.08     RATE / HR.                 106.2
    TCB TIME                0.07        0.07     NORMAL TERMINATIONS           23
    SRB TIME                0.02        0.01     ABNORMAL TERMINATIONS          0
                                                 SQL QUERY/UPDATE CALLS       5.2
                                                 LOCK SUSPENSIONS             0.0

                         ---------- AVERAGES PER # OCCUR. ----------  GRAND
BUFFER MANAGER SUMMARY    POOL 0   POOL 1   POOL 2   POOL 32  TOTAL    TOTAL
------------------------ -------- -------- -------- -------- -------- --------
GET PAGE REQUEST             68.6      0.0      0.0      0.0     68.6     1580
BUFFERPOOL EXPANSIONS         0.0      0.0      0.0      0.0      0.0        0
SYSTEM PAGE UPDATES          15.6      0.0      0.0      0.0     15.6      359
SYNCHRONOUS READ I/O          2.4      0.0      0.0      0.0      2.4       56
SEQ. PREFETCH REQUEST         3.1      0.0      0.0      0.0      3.1       72

LOCKING SUMMARY            TOTALS    SUSPENSIONS (CLASS 3)        AVERAGE TIME
-------------------------- --------  ---------------------        ------------
SUSPENSIONS                       0  I/O SUSPENSIONS                      0.10
DEADLOCKS                         0  LOCK/LATCH SUSPENSIONS               0.00
TIMEOUTS                          0
LOCK ESCALATION (SHARED)          0
LOCK ESCALATION (EXCLUS)          0
MAXIMUM PAGE LOCKS HELD          87
ACCOUNTING INVOCATIONS
-------------------------
NORMAL
   NEW USER                       0
   DEALLOCATION                  23
   APPL.PROGRAM END               0

ABNORMAL
   APPL.PROGRAM ABEND             0
   END OF MEMORY                  0
   RESOLVE IN DOUBT               0
   CANCEL FORCE                   0

WORK UNIT IN DOUBT
   APPL.PROGRAM ABEND             0
   END OF TASK                    0
   END OF MEMORY                  0
   RESOLVE IN DOUBT               0
   CANCEL FORCE                   0
```

Figure 72 (Part 1 of 2). Accounting Detail Report for a CICS Transaction

tables initiated by remote requesters that all have the same PLANAPP1 plan name at the server site.

- PACKAGE is used to identify a package regardless of the plan to which it belongs. When usage is reported on a per package basis, it is not possible to attribute activity to specific plans or other DB2PM identifiers.

Highlights of the additional new functions are:

```
                                          TOTAL/
MANIPULATIVE SQL ACTIVITY   TOTAL    #OCCUR
------------------------- --------  --------
QUERY/UPDATE ACTIVITY
     SELECT                     98       4.2
     INSERT                      4       0.1
     UPDATE                     15       0.6
     DELETE                      4       0.1
     PREPARE                     0       0.0

OTHER ACTIVITY
     DESCRIBE                    0       0.0
     OPEN                       17       0.7
     CLOSE                      17       0.7
     FETCH                      96       4.1
     COMMIT                     22       0.9
     ABORT                       0       0.0

CONTROL SQL ACTIVITY        TOTAL
-------------------------   ------
LOCK TABLE                       0
GRANT                            0
REVOKE                           0
INCR. BINDS                      0
COMMENT                          0
LABEL                            0

DEFINITIONAL SQL ACTIVITY   CREATE  DROP  ALTER
-------------------------   ------ ------ ------
TABLE                            0      0      0
INDEX                            0      0      0
TABLESPACE                       0      0      0
STORAGE GROUP                    0      0      0
DATABASE                         0      0
SYNONYM                          0      0
VIEW                             0      0
END OF REPORT.
```

Figure 72 (Part 2 of 2). Accounting Detail Report for a CICS Transaction

- Accounting reports and traces have a repeating group of fields for each package or DBRM that was referenced during the duration of the accounting record (see Figure 73 on page 142).
- INCLUDE and EXCLUDE are expanded to support specific processing for packages or DBRMs.
- ORDER is expanded to support the ordering of accounting reports by PACKAGE (see Figure 74 on page 143) and MAINPACK (see Figure 76 on page 144).
- The SQL activity trace and report can be summarized by program, and the program can be either a DBRM or a package.
- Record trace provides all package-related data captured by the DB2 instrumentation facility.
- Exception processing (batch and online) supports the new package fields and allows qualification of exceptions by package name.
- Online monitor thread displays contain the new package fields for the current package.

Figure 73 on page 142 shows a sample of an accounting short trace where two packages are involved. The plan name is PLANAPP1.

```
1
0 ============================================================
0                ACCOUNTING
0                  TRACE
0                    LAYOUT(SHORT)
1   LOCATION: CICSLOC                          DB2 PERFORMANCE MONITOR (V3)                              PAGE:
1-1
   SUBSYSTEM: DSN2                               ACCOUNTING TRACE - SHORT                        DB2 VERSION: V3
 ACTUAL FROM: 09/08/93 04:11:42.31                                                            REQUESTED FROM: ALL
04:11:00.00
   PAGE DATE: 09/08/93                                                                                 TO: DATES
04:15:00.00

 PRIMAUTH CORRNAME CONNECT   ACCT TIMESTAMP COMMITS   OPENS UPDATES INSERTS  EL. TIME(CL1)  EL. TIME(CL2) GETPAGES SYN.READ LOCK SUS
 PLANNAME CORRNMBR THR.TYPE TERM. CONDITION SELECTS FETCHES DELETES PREPARE  TCB TIME(CL1)  TCB TIME(CL2) BUF.UPDT TOT.PREF LOCKOUTS
 -------- -------- -------- ---------------- ------- ------- ------- ------- -------------- -------------- -------- -------- --------

 USRT001  004C0001 CICS410  04:11:42.312608       2       1       0       0       5.974271       0.120076       12       10        1
 PLANAPP1 'BLANK'  ALLIED   NORM DEALLOC          0      65       0       0       0.027459       0.024048        0        1        0

     ---------------------------------------------------------------------------------------
     |PROGRAM NAME        TYPE      SQLSTMT  CL7 ELAP.TIME   CL7 TCB TIME  CL8 SUSP.TIME  CL8 SUSP|
     |SSQLFFO             PACKAGE        68       0.119931       0.023910       0.090771         8|
     ---------------------------------------------------------------------------------------


 USRT001  004D0001 CICS410  04:12:11.647066       2       0       0      64       5.872581       0.664971      151       18        1
 PLANAPP1 'BLANK'  ALLIED   NORM DEALLOC          0       0       0       0       0.157637       0.103878      197        0        0

     ---------------------------------------------------------------------------------------
     |PROGRAM NAME        TYPE      SQLSTMT  CL7 ELAP.TIME   CL7 TCB TIME  CL8 SUSP.TIME  CL8 SUSP|
     |SSQL                PACKAGE        64       0.664814       0.103727       0.554440        18|
     ---------------------------------------------------------------------------------------


 USRT001  004E0001 CICS410  04:12:40.832168       2       1      64       0       7.827162       0.189291       17        3        1
 PLANAPP1 'BLANK'  ALLIED   NORM DEALLOC          0      65       0       0       0.140760       0.077407       64        1        0

     ---------------------------------------------------------------------------------------
     |PROGRAM NAME        TYPE      SQLSTMT  CL7 ELAP.TIME   CL7 TCB TIME  CL8 SUSP.TIME  CL8 SUSP|
     |SSQL                PACKAGE       132       0.189144       0.077266       0.110461         4|
     ---------------------------------------------------------------------------------------


 USRT001  004F0001 CICS410  04:13:12.425355       2       1       0       0       8.651578       0.222817      342        3        1
 PLANAPP1 'BLANK'  ALLIED   NORM DEALLOC          0      65      64       0       0.141431       0.098112      197        1        0

     ---------------------------------------------------------------------------------------
     |PROGRAM NAME        TYPE      SQLSTMT  CL7 ELAP.TIME   CL7 TCB TIME  CL8 SUSP.TIME  CL8 SUSP|
     |SSQL                PACKAGE       132       0.222658       0.097969       0.123500         4|
     ---------------------------------------------------------------------------------------


 ACCOUNTING TRACE COMPLETE
```

Figure 73. DB2PM Accounting Short Trace

Figure 74 on page 143 shows the accounting short report ordered by package. The report indicates the use of resources by package, regardless of the plan under which a particular package is executed.

```
1
0 ============================================================
0                ACCOUNTING
0                  REPORT
0                    LAYOUT(SHORT)
0                    ORDER(PACKAGE)
1     LOCATION: CICSLOC                          DB2 PERFORMANCE MONITOR (V3)                              PAGE:
1-1
     SUBSYSTEM: DSN2                              ACCOUNTING REPORT - SHORT                      DB2 VERSION: V3
 INTERVAL FROM: 09/08/93 04:11:42.31                                                          REQUESTED FROM: NOT SPECIFIED
           TO: 09/08/93 04:13:12.42                      ORDER: PACKAGE                                 TO: NOT SPECIFIED

                                                          TYPE            SQLSTMT   CL7 TCB TIME  CL8 SUSP
 PACKAGE                                                   #OCCURS  CL7 ELAP.TIME  CL8 SUSP.TIME
 ----------------------------------------------------------  --------  -------------  -------------  --------

 CICSLOC.USRT001.SSQL                                     PACKAGE          109.33       0.092988      8.67
                                                                 3       0.358872       0.262800

 CICSLOC.USRT001.SSQLFFO                                  PACKAGE           68.00       0.023910      8.00
                                                                 1       0.119931       0.090771

 ***  GRAND TOTAL  ***
                                                          PACKAGE           99.00       0.075718      8.50
                                                                 4       0.299137       0.219793

 ACCOUNTING REPORT COMPLETE
```

Figure 74. DB2PM Accounting Short Report Ordered by PACKAGE

Figure 75 shows the accounting short report ordered by plan name. Because the plan name is PLANAPP1, data for different packages is summarized under this plan.

```
1
0 ============================================================
0                ACCOUNTING
0                  REPORT
0                    LAYOUT(SHORT)
0                    ORDER(PLANNAME)
1     LOCATION: CICSLOC                          DB2 PERFORMANCE MONITOR (V3)                              PAGE:
1-1
     SUBSYSTEM: DSN2                              ACCOUNTING REPORT - SHORT                      DB2 VERSION: V3
 INTERVAL FROM: 09/08/93 04:11:42.31                                                          REQUESTED FROM: NOT SPECIFIED
           TO: 09/08/93 04:13:12.42                      ORDER: PLANNAME                                TO: NOT SPECIFIED

                           #OCCURS #ROLLBK SELECTS INSERTS UPDATES DELETES CLASS1 EL.TIME CLASS2 EL.TIME GETPAGES SYN.READ LOCKSUS
 PLANNAME                  #DISTRS #COMMIT FETCHES   OPENS  CLOSES PREPARE CLASS1 TCBTIME CLASS2 TCBTIME BUF.UPDT TOT.PREF #LOCKOUT
 ------------------------- ------- ------- ------- ------- ------- ------- -------------- -------------- -------- -------- --------

 PLANAPP1                        4       0    0.00   16.00   16.00   16.00       7.081398       0.299289   130.50     8.50   1.00
                                 4       8   48.75    0.75    0.75    0.00       0.116822       0.075861   114.50     0.75   0

     ------------------------------------------------------------------------------------------------
     |PROGRAM NAME       TYPE      #OCCURS  SQLSTMT  CL7 ELAP.TIME   CL7 TCB TIME  CL8 SUSP.TIME  CL8 SUSP|
     |SSQL               PACKAGE         3   109.33       0.358872       0.092988       0.262800      8.67|
     |SSQLFFO            PACKAGE         1    68.00       0.119931       0.023910       0.090771      8.00|
     ------------------------------------------------------------------------------------------------


 ACCOUNTING REPORT COMPLETE
```

Figure 75. DB2PM Accounting Short Report Ordered by Plan Name

Figure 76 shows the accounting report ordered by main package within a plan name. This report facilitates distinguishing between accounting records that have the same plan name but executed different packages. Thus, the break up of the two packages SSQL and SSQLFFO executed under the same plan PLANAPP1 is presented. This would not be possible if the report were ordered by plan name as illustrated in Figure 75 on page 143.

```
1
0 ============================================================
0                 ACCOUNTING
0                   REPORT
0                     LAYOUT(SHORT)
0                     ORDER(PLANNAME-MAINPACK)
1     LOCATION: CICSLOC                          DB2 PERFORMANCE MONITOR (V3)                          PAGE:
1-1
     SUBSYSTEM: DSN2                               ACCOUNTING REPORT - SHORT                     DB2 VERSION: V3
 INTERVAL FROM: 09/08/93 04:11:42.31                                                          REQUESTED FROM: NOT SPECIFIED
           TO: 09/08/93 04:13:12.42                 ORDER: PLANNAME-MAINPACK                              TO: NOT SPECIFIED

 PLANNAME                  #OCCURS #ROLLBK SELECTS INSERTS UPDATES DELETES CLASS1 EL.TIME CLASS2 EL.TIME GETPAGES SYN.READ LOCKSUS
  MAINPACK                 #DISTRS #COMMIT FETCHES   OPENS  CLOSES PREPARE CLASS1 TCBTIME CLASS2 TCBTIME BUF.UPDT TOT.PREF #LOCKOUT
 ------------------------- ------- ------- ------- ------- ------- ------- -------------- -------------- -------- -------- --------

 PLANAPP1                        3       0    0.00   21.33   21.33   21.33       7.450440       0.359026   170.00     8.00    1.00
  SSQL                           3       6   43.33    0.67    0.67    0.00       0.146609       0.093132   152.67     0.67    0

    ----------------------------------------------------------------------------------------------------
    |PROGRAM NAME        TYPE      #OCCURS  SQLSTMT  CL7 ELAP.TIME   CL7 TCB TIME  CL8 SUSP.TIME  CL8 SUSP|
    |SSQL                PACKAGE         3   109.33       0.358872       0.092988       0.262800      8.67|
    ----------------------------------------------------------------------------------------------------


 PLANAPP1                        1       0    0.00    0.00    0.00    0.00       5.974271       0.120076    12.00    10.00    1.00
  SSQLFFO                        1       2   65.00    1.00    1.00    0.00       0.027459       0.024048     0.00     1.00    0

    ----------------------------------------------------------------------------------------------------
    |PROGRAM NAME        TYPE      #OCCURS  SQLSTMT  CL7 ELAP.TIME   CL7 TCB TIME  CL8 SUSP.TIME  CL8 SUSP|
    |SSQLFFO             PACKAGE         1    68.00       0.119931       0.023910       0.090771      8.00|
    ----------------------------------------------------------------------------------------------------


 ***  TOTAL  ***
 PLANAPP1                        4       0    0.00   16.00   16.00   16.00       7.081398       0.299289   130.50     8.50    1.00
                                 4       8   48.75    0.75    0.75    0.00       0.116822       0.075861   114.50     0.75    0

    ----------------------------------------------------------------------------------------------------
    |PROGRAM NAME        TYPE      #OCCURS  SQLSTMT  CL7 ELAP.TIME   CL7 TCB TIME  CL8 SUSP.TIME  CL8 SUSP|
    |ALL PROGRAMS        PACKAGE         4    99.00       0.299137       0.075718       0.219793      8.50|
    ----------------------------------------------------------------------------------------------------


 ACCOUNTING REPORT COMPLETE
```

Figure 76. DB2PM Accounting Short Report Ordered by MAINPACK within Plan Name

7.1.4.5 Using the DB2 Performance Facility

The DB2 performance facility trace provides detailed information on the flow of control inside DB2. While the main purpose of this trace is to supply debugging information, it can also be used as a monitoring tool because of the timing data provided with each entry.

Due to high resource consumption, the DB2 performance trace should be used only in specific cases, where it becomes difficult to use any other tool to monitor DB2-oriented transactions.

Even in this case, only the needed classes of performance trace should be started, for only a limited time and for only the transactions that need to be carefully monitored.

7.1.4.6 Monitoring Thread Reuse

From a performance viewpoint it is important to be able to reuse a thread. Thread reuse can be estimated as:

thread reuse % = ((commits - threads_used)/commits) x 100 %

The thread reuse % is close to one hundred if few threads are used, and zero if there is one thread created for each commit.

For terminal-oriented transactions the thread is released for each commit. One transaction can then be associated with more than one thread. The number of COMMITs then expresses the number of times that a thread is allocated to a transaction.

For non-terminal-oriented transactions the thread is not released at commit, but only at thread termination. The commit count should then be adjusted to one for each transaction before using the formula.

Monitoring thread reuse can be performed at different levels:

DB2 level
CICS level
Plan level for each CICS system.

For performance follow-up, it is normally necessary to measure thread reuse at the plan level for each CICS system.

The following DB2PM command produces an accounting report, from which the number of commits and the number of threads can be taken:

```
DB2PM ACCOUNTING (REDUCE,REPORT(LEVEL(DETAIL),ORDER(order)))
```

The fields COMMIT and DEALLOCATION in the accounting report describe the number of commits and the number of plan deallocations that can be used for the number of threads used.

It is assumed that the input to this report covers a full day, because the accounting records may not be written out before the thread is terminated. For threads that are reused, this can be at the end of the day.

If “order” in the above DB2PM command is set to “CONNECTION,” one accounting report is produced for each CICS system. Thread reuse can then be calculated for each system.

If “order” is set to “CONNECTION-PLANNAME” one report is produced for each plan in each CICS system. Thread reuse can then be calculated at the plan level.

Both reports produce a “Grand Total” report that can be used to calculate the thread reuse for all CICS systems at the DB2 level. The DB2PM EXCLUDE option can be used to exclude non-terminal-oriented transactions from the DB2PM reports. This can be useful, if there are many commits in these transactions and the above formula is used to calculate the thread reuse.

7.1.5 Monitoring CICS

CICS provides monitoring and accounting facilities for the resources needed by CICS transactions within the CICS address space. Three types of records can be produced:

- Performance records that record the resources used by each transaction in the CICS address space. DL/I calls are also recorded.
- Exception records that record shortages of resources
- Event records for transaction tracing using the Resource Measurement Facility (RMF) capabilities.

CICS monitoring is used in the CICS-DB2 environment, together with the DB2 accounting facility, to monitor performance and to collect accounting information.

CICS also provides statistics records to monitor CICS as a whole. Information about the CICS system behavior can be found in these reports. Refer to *CICS/ESA Customization Guide* and *CICS/ESA Performance Guide* for a details on CICS monitoring and statistics.

7.1.6 Monitoring the Overall MVS System

We do not cover RMF usage in this chapter. We mention it here as a way to obtain processor times for CICS and both DB2 address spaces. At MVS system level, this method provides processor times as well as I/O and paging information.

RMF processor times can be used for accounting if a distribution formula based on resources used by the transactions (that is, SQL statements, accesses to DL/I, and so on) is supplied. See Chapter 8, “Accounting” on page 153.

7.2 Tuning

As mentioned previously, this guide does not cover tuning DB2 tables and the DB2 subsystem. For information on these topics and for general considerations when tuning a DB2 application, refer to *IBM DATABASE 2 Administration Guide*.

7.2.1 Tuning a CICS Application

Tuning a CICS application must be done in two phases:

- Before moving an application to production
- On a periodical basis when in production.

When moving an application to production, add these checks to those already performed for CICS:

- Ensure that the number and type of SQL statements used meet the program specifications (use the DB2 accounting facility).
- Check if the number of get and updated pages in the buffer pool is higher than expected (use the DB2 accounting facility).
- Check that planned indexes are being used (use EXPLAIN), and that inefficient SQL statements are not being used.
- Check if DDL is being used and, if so, the reasons for using it (use the DB2 accounting facility).

- Check if conversational transactions are being used.

  Determine whether pseudoconversational transactions can be used instead. If conversational design is needed, check the DB2 objects that will be locked across conversations. Check also that the number of new threads needed because of this conversational design is acceptable.

- Check the locks used and their duration.

  Make sure that tablespace locks are not being used because of incorrect or suboptimal specification of, for example:

  – LOCK TABLE statement

  – LOCKSIZE=TS specification

  – ISOLATION LEVEL(RR) specification

  – Lock escalation.

  This information is available in the catalog tables, except for lock escalation, which is an installation parameter (DSNZPARM).

- Check the plans used and their sizes. Even though the application plans are segmented, the more DBRMs used in the plan, the longer the time needed to BIND and REBIND the plans in case of modification. Try to use packages whenever possible. Packages were designed to solve the problems of:

  – Binding the whole plan again after modifying your SQL application. (This was addressed by dynamic plan selection, at the cost of performance.)

  – Binding each application plan if the modified SQL application is used by many applications.

When this tuning is complete, use the expected transaction load to decide on the type of entries to define in the RCT and the number of threads required. Check also the impact of these transactions on the DB2 and CICS subsystems.

When tuning an application in production:

- Check that the CICS applications use the planned indexes by monitoring the number of GET PAGES in the buffer pool (use the DB2 accounting facility). The reasons for an index not being used may be that the index has been dropped, or that the index was created after the plan was bound.
- Use the lock manager data from the accounting facility to check on suspensions, deadlocks, and timeouts.

7.2.2 Tuning the CICS Attachment Facility

When tuning the CICS attachment facility, you must consider the changes to CICS architecture due to the connection to DB2.

- The CICS main TCB uses its internal dispatching to service CICS tasks. However, when a program issues an SQL statement, the CICS task goes into a CICS WAIT state, and DB2 code runs under a different subtask-TCB, the thread TCB. Each DB2 thread is served by one TCB.
- Parallel processing can occur both between the CICS main TCB and thread TCBs, and between thread TCBs.
- There is no multithreading under a thread TCB. A thread services only one request at a time.

- A page fault under the CICS main TCB means that all processing under the main CICS TCB stops. A page fault under a thread TCB does not affect CICS, DB2, or other thread TCBs.

All the recommendations concerning tuning the CICS attachment facility relate to RCT parameter definition. They are explained in detail in Chapter 5, "Defining the Resource Control Table" on page 75.

In summary, the objectives in tuning the CICS attachment facility are to:

- Optimize the number of threads in the connection.

  The total number of threads in the connection, and the number of threads for each dedicated entry and the pool must be optimized. A larger number of threads than is needed requires additional processor time to dispatch the TCBs and additional storage for plans, data, and control blocks. If an insufficient number of threads is defined in the RCT, response time increases.

- Optimize the assignment and reuse of threads.

  Reusing threads avoids the thread creation and termination process, including plan allocation and authorization checks. Thread creation and termination represent a significant part of the processing time for a simple transaction.

  Limit conversational transactions either through transaction classes or by using a dedicated entry (THRDA greater than 0) with TWAIT=YES specified. Otherwise, they tie up the pool. Do not allow conversational transactions to use the pool.

- Choose the priority assigned to the thread subtasks, using the DPMODI or DPMODE parameter.

- Choose the best authorization strategy to avoid or minimize the process of sign-on by each thread.

7.3 Handling Deadlocks

Deadlocks can occur in a CICS-DB2 system between two or more transactions or between one transaction and another DB2 user.

This section covers deadlocks only within DB2. If DB2 resources are involved in this type of deadlock, one of the partners in the deadlock times out according to the user-defined IRLM parameters. Other possible deadlocks are where resources outside DB2 are involved.

Deadlocks are expected to occur, but not too often. You should give special attention to deadlock situations if:

- Other transactions are often delayed because they access resources held by the partners in the deadlock. This increases the response times for these transactions. A cascade effect can then be the result.

- The resources involved in the deadlock are expected to be used more intensively in the future, because of an increased transaction rate either for the transactions involved in the deadlock or for other transactions.

The IRLM component of the DB2 subsystem performs deadlock detection at user-defined intervals. One of the partners in the deadlock is the victim and

receives a -911 or a -913 return code from DB2. The actual return code is determined by the ROLBE parameter for this transaction's entry in the RCT. The other partner continues processing after the victim is rolled back.

A more detailed discussion of the return codes and the recommended programming actions are given in the section 5.5.3.5, "ROLBE" on page 88.

To solve deadlock situations, you must perform a number of activities. Solving deadlocks means applying changes somewhere in the system to reduce the deadlock likelihood.

The following steps are often necessary for solving a deadlock situation:

1. Detect the deadlock.
2. Find the resources involved.
3. Find the SQL statements involved.
4. Find the access path used.
5. Determine why the deadlock occurred.
6. Make changes to avoid it.

7.3.1 Two Deadlock Types

A deadlock within DB2 can occur when two transactions are both holding a lock wanted by the other transaction. In a DB2 environment, two deadlock types can occur when:

- Two resources are involved.
  Each transaction has locked one resource and wants the other resource in an incompatible mode. The resources are typically index pages and data pages. This is the classic deadlock situation.
- Only one resource is involved.
  DB2 Release 2 introduced the concept of update (U) locks. The main purpose was to reduce the number of situations where a *lock promotion* caused the deadlock. The U-lock has solved most of these situations, but it is still possible in specific situations to have a deadlock with only one resource involved, because the resource can be locked in more than one way.

  A typical example of this is when a transaction opens a cursor with the ORDER BY option and uses an index to avoid the sort. When a row in a page is fetched, DB2 takes a share (S) lock at that page. If the transaction then issues an update without a cursor for the row last fetched, the S-lock is promoted to an exclusive (X) lock.

  If two of these transactions run concurrently and both get the S-lock at the same page before taking the X-lock, a deadlock occurs.

7.3.2 Deadlock Detection

In a normal production environment running without DB2 performance traces activated, the easiest way to get information about a deadlock is to scan the MVS log to find the messages shown in Figure 77 on page 150.

```
DSNT375I PLAN p1 WITH CORRELATION ID id1
AND CONNECTION ID id2 IS DEADLOCKED with
PLAN p2 WITH CORRELATION ID id3
AND CONNECTION ID id4.

DSNT501I DSNILMCL RESOURCE UNAVAILABLE
         CORRELATION-ID=id1,CONNECTION-ID=id2
         REASON=r-code
         TYPE name
         NAME name
```

Figure 77. Deadlock Messages. The messages identify the transactions involved in the deadlock.

From these messages, both partners in the deadlock are identified. The partners are given by both plan-name and correlation ID.

Also, a second message identifies the resource that the victim could not obtain. The other resource (whether it is the same or not) is not displayed in the message.

7.3.3 Finding the Resources Involved

To find the other resources involved in a deadlock, you may have to activate a DB2 performance trace and re-create the deadlock. Suppose that the reason for solving the deadlock is that the number of deadlocks is too high. Normally re-creating the deadlock after the trace is started is a minor problem.

You should limit the DB2 performance trace to the two plans indicated in the MVS log message. The ″AUTH RCT″ parameter specifies the CICS transaction ID; so limiting the trace to the two transaction IDs (authorization IDs) involved can also be reasonable. The performance trace to be started should include *class(06)* for general locking events and *class(03)* for SQL events. The Database 2 Performance Monitor (DB2PM) is a useful tool to format the trace output. The DB2PM lock contention report and the lock suspension report can assist in determining the resources involved in the deadlock.

If the output from the DB2PM reports is too large, you can develop a user program to analyze the output from the traces. The goal is to find the resources involved in the deadlock and all the SQL statements involved.

7.3.4 Finding the SQL Statements Involved

A deadlock can involve many SQL statements. Often solving the deadlock requires finding all SQL statements. If the resources involved are identified from the lock traces, you can find the involved SQL statements in an SQL trace report by combining the timestamps from both traces.

7.3.5 Finding the Access Path Used

To find the access path used by the SQL statements involved in the deadlock, use the EXPLAIN option of DB2 for the corresponding plans.

7.3.6 Determining Why the Deadlock Occurred

Identifying both the SQL statements and the resources involved in the deadlock and finding the access path should show you why the deadlock occurred. This knowledge is often necessary to be able to develop one or more solutions. However, the process can be time-consuming.

7.3.7 Making Changes

In general, a deadlock occurs because two or more transactions both want the same resources in opposite order at the same time and in a conflicting mode. The actions taken to prevent a deadlock must deal with these characteristics.

Figure 78 shows a list of preventive actions and the corresponding main effects.

Actions	Spread Resources	Change the Locking Order	Decrease Concurrency	Change Locking Mode
Increase Index Freespace	X			
Increase Index Subpage size	X			
Increase TS Freespace	X			
Change Clustering Index	X	X		
Reorg the Table Space	X	X	X	
Add an Index		X	X (1)	
Drop an Index		X		
Serialize the Transactions			X	
Use additional COMMITS			X	
Minimize the Response Time			X	
Change Isolation Level (2)			X	X
Redesign Application	X	X	X	X
Redesign Database	X	X	X	X

(1) Due to changes in access path.
(2) Cursor Stability is usually better than repeatable read.

Figure 78. Deadlock Prevention

To choose the right action, you must first understand why the deadlock occurred. Then you can evaluate the actions to make your choices. These actions can have several effects. They can:

- Solve the deadlock problem as desired
- Force a change in access path for other transactions causing new deadlocks
- Cause new deadlocks in the system.

It is therefore important that you carefully monitor the access path used by the affected transactions, for example by the EXPLAIN facility in DB2. In many cases, solving deadlocks is an iterative process.

Chapter 8. Accounting

Accounting in a CICS environment can be done to:

- Charge back the total amount of resources consumed for a given set of transactions to a well-defined set of end users.
- Analyze the transactions being executed in the system.

Normally the units of consumption are the processor, I/O, main storage, and so on, in some weighted proportion. A typical CICS transaction consumes resources in the:

Operating system
CICS system
Application code
DB2 address spaces.

Each of these components can produce data, which can be used as input to the accounting process. It is the user's responsibility to combine the output from the different sources.

A normal requirement of an accounting procedure is that the results calculated are repeatable, which means that the cost of a given transaction accessing a given set of data should be the same whenever the transaction is executed. In most cases, this means that the input data to the accounting process should also be repeatable.

This chapter discusses only the data delivered by CICS and DB2.

For simplification, the term end user is used in this chapter as the target for charging resources. The end user can be real end users, groups of end users, transactions, or any other expression for the unit to which the resources must be appointed.

8.1 CICS-Supplied Information

Several facilities are included in CICS to perform the tasks of accounting and performance. These facilities can be used to measure the use of different resources within a CICS system. The most frequently used tools are:

- *Statistics* data. CICS statistics are the simplest tool for permanently monitoring a CICS system. They contain information about the CICS system as a whole, such as its performance and use of resources. This makes CICS statistics suitable for performance tuning and capacity planning. Statistics are collected during CICS online processing and are processed offline later. The statistics domain collects this data and then writes records to the System Management Facility (SMF) dataset provided by MVS. The records are of the SMF type 110. These records can be processed offline by the DFHSTUP program.

- *Monitor* data. CICS monitoring collects data about all user and CICS-supplied transactions during online processing for later offline analysis. The records produced by CICS monitoring are also the SMF type 110 and are written to the SMF data sets. Data provided by CICS monitoring is useful for

performance tuning and for charging your users for the resources they use. Monitoring provides three classes of data:

- Performance class for detailed transaction level information
- Exception class for exceptional conditions
- Sysevent class for special kind of transaction information.

For a detailed description of the CICS monitoring facilities, see the *CICS/ESA Customization Guide* and the *CICS/ESA Performance Guide*. Refer to this documentation for details on activating, collecting, and processing this information.

For both Statistics data and Monitor data, an offline processing facility can be used. Service Level Reporter (SLR) is one of the tools that collects and analyzes data from CICS and other IBM systems and products. Examples provided by CICS in the *CICS/ESA Customization Guide* show how to manipulate the SMF records. SLR can build reports that help you with:

- Systems overview
- Service levels
- Availability
- Performance and tuning
- Capacity planning
- Change and problem management
- Accounting.

8.2 DB2-Supplied Information

The instrumentation facility component of DB2 offers you the possibility to use six types of traces. For each trace type, you can activate a number of trace classes. You can use SMF as the trace output destination. Another alternative is to externalize the trace output under control of GTF. The types of traces are statistics, accounting, audit, performance, monitor, and global.

Statistics Describe the total work executed in DB2. This information is not related to any specific end user. The main purposes of the DB2 statistics trace are to:

- Supply data for DB2 capacity planning
- Assist with monitoring and tuning at the DB2 subsystem level
- Assist in accounting for DB2 activity.

The statistics records are written at user-defined intervals. You can reset the statistical collection interval and the origination time without stopping and starting the trace by using the MODIFY TRACE command. All DB2 activity for the statistical collection interval is reported in the record. This makes it difficult to directly relate the activity to specific end users.

The DB2 statistics trace can be activated for several classes. If the statistics records are written to SMF, the SMF types are 100 and 102.

Accounting Describes the work performed on behalf of a particular user (authorization ID from the RCT table). The main purposes of the accounting records are to charge the DB2 cost to the authorization ID and perform monitoring and tuning at the program level. DB2 produces an accounting record at thread termination or when a transaction is reusing a thread with a new authorization ID. That means that if a thread is defined as protected (THRDS>0) and all transactions with the same transaction code for this RCT entry use the same authorization ID, only one accounting record is produced, describing all activity done in the thread. Additionally, accounting records are written if you use TOKENE(I)=YES in your RCT definitions. This is considered a sign-on, even if you use the same authorization ID.

With CICS/ESA Version 4 a new parameter, TXIDSO, was added to the RCT and affects the sign-on process of the thread. Refer to Chapter 5, “Defining the Resource Control Table” on page 75 for an explanation of this new facility.

You can activate the DB2 accounting trace for several classes. If the accounting records are written to SMF, the SMF type is 101 and 102.

Audit Collects information about DB2 security controls and is used to ensure that data access is allowed only for authorized purposes. If the audit records are written to SMF, the SMF type is 102.

Performance Records information for a number of different event classes. The information is intended for:

- Program-related monitoring and tuning
- Resource-related monitoring and tuning
- User-related monitoring and tuning
- System-related monitoring and tuning
- Accounting-related profile creation.

You can activate the DB2 performance trace for several classes. If the performance records are written to SMF, the SMF type is 102.

Monitor Records data for *online* monitoring with user written programs

Global Aids serviceability. If the global trace records are written to SMF, the SMF type is 102.

8.3 Accounting for Processor Usage

In this section we provide information on the reporting of processor resources used by CICS and DB2. We also give details about the different accounting classes in the DB2 accounting record.

8.3.1 CICS-Generated Processor Usage Information

CICS is a multitasking address space, and CICS monitoring facilities are generally used to determine the processor time and other resources consumed by the individual transactions or functions performed by CICS on its behalf.

In CICS/ESA 3.3, for example, data in the dispatcher statistics section provides the accumulated time for each of the CICS TCBs, the quasi-reentrant TCB, the resource-owning TCB, the concurrent TCB (this is optional and depends on the SUBTSKS=0|1 SIT parameter), and the FEPI TCB (present if FEPI=YES in SIT). The field Accum/TCB is the total processor time used by the corresponding TCB. Check the documentation provided for CICS/ESA 4.1 and subsequent releases.

You can obtain the total processor time used by the CICS address space from RMF workload (WLKD) reports. This time can be greater than the sum of all CICS task TCBs.

The difference between the processor time reported by RMF and the sum of CICS TCBs in the CICS dispatcher statistics report is the processor time consumed by all the other subtask TCBs. The subtasks are used for:

- DB2 threads
- Journaling tasks, such as find end of volume, open, close, and switch of CICS journals
- File related tasks, such as allocation, open, close, and deallocation of CICS application data sets defined in the file control table (FCT) or RDO file definitions
- If RACF is present, requests to RACF that require physical I/O
- If DL/I is present, database allocation, open, close, and deallocation
- If DBCTL is used, processor time consumed in the DBCTL threads.

These are global performance reports and can help you determine how much of your processor time is being used by CICS.

In CICS regions, DB2 thread TCBs always use MVS subtasks separately from the CICS main task TCB. Consequently, you need more granularity to get the proper results.

You can obtain transaction performance records from SLR reports. SLR reads the SMF datasets searching for SMF type 110 records, in the case of CICS, and can provide a matrix, tabular, or any graphics for each transaction run in the CICS system. This is accomplished by using two CICS SLR tables: CICSTRANSUM and CICSTRANSLOG. Monitoring data from CICS systems can be processed to obtain, for example:

- Transaction response time
- Terminal control I/O and wait count
- Temporary storage I/O and wait count
- File control I/O and wait count
- Transaction suspend count and time
- Transaction rate
- Journal control I/O and wait count
- Journal control requests and I/O count
- Transient data request
- Processor utilization
- BMS requests.

Figure 79 on page 157 could be a row in a tabular SLR report.

```
TRAN STOP...... RESP  OTHER CPU  DISP  FC   TC   JC   TS
     TIME            WAIT       TIME WAIT WAIT WAIT WAIT
---- ---------- ----- ----- ---- ---- ---- ---- ---- --------> OTHER
XC05 19:09:35.7 2.774 2.765 .008 .009 .000 .000 .013 .000
```

Figure 79. Sample of SLR output

In this report wait time on DB2 TCB is reported in the field OTHER WAIT. That is the time you must monitor when trying to find any abnormal condition in your DB2 applications. If this value is very high, then further investigation in DB2 is necessary.

SLR allows you to customize your report to show other information (for example, the transaction start time instead of the transaction stop time), or to show the information in Figure 79 in a different order.

8.3.2 DB2-Generated Processor Usage Information

This section covers the information about processor time consumption provided in the DB2 accounting and statistics trace records. The subsequent sections give details about the processor times supplied by each DB2 accounting class.

The DB2 *accounting trace* can be started with CLASS 1, CLASS 2 or CLASS 3. However, CLASS 1 must always be active to externalize the information collected by activating CLASS 2, CLASS 3, or both classes.

The processor times reported in the accounting records are the TCB time for the thread TCB running code in CICS or in the DB2 address space using cross-memory services and the SRB time for work scheduled in CICS.

CLASS 1 (the default) results in accounting data being accumulated by several DB2 components during normal execution. This data is then collected to write the DB2 accounting record. The data collection does not involve any overhead of individual event tracing.

CLASS 2 and CLASS 3 activate many additional trace points. Every occurrence of these events is traced internally, but is *not* written to an external destination. Rather, the accounting facility uses these traces to compute the additional total statistics that appear in the accounting record when CLASS 2 or CLASS 3 is activated. Accounting CLASS 1 must be active to externalize the information.

CLASS 2 collects the delta elapsed and processor times spent ′IN DB2′ and records this in the accounting record.

CLASS 3 collects the I/O elapsed time and lock and latch suspension time spent ′IN DB2′ and records this in the accounting record.

CLASS 7 and CLASS 8 in DB2 Version 3 collect package level accounting in DB2 and package level accounting wait in DB2. For information on package level accounting, refer to *IBM DATABASE 2 Administration Guide*.

The *statistics trace* reports processor time in the statistics records. The processor times reported are:

- Time under a DB2 address space TCB running asynchronous of the CICS address space. Examples of this are the DB2 log and writes from the buffer pools.
- Time under SRBs scheduled under the DB2 address spaces. An example is the asynchronous read engine for sequential prefetch.

The DB2 address spaces reported in the statistics record are:

- Database manager address space
- System services address space
- IRLM.

In a CICS environment, the processor time from the DB2 accounting records is typically much greater than the processor time reported in the DB2 statistical records, because most of the processor time used is in the thread TCB itself and in the DB2 address spaces using cross memory services.

8.3.2.1 Accounting CLASS 1 Processor Time

For CLASS 1, a task processor timer is created when the TCB is attached. When a thread to DB2 starts, the timer value is saved. When the thread is terminated (or the authorization ID is changed), then the timer is checked again, and both the timer start and end values are recorded in the SMF type 101 record. The fields in the SMF 101 record used for accounting CLASS 1 processor time are:

- QWACBJST for begin thread TCB time
- QWACEJST for end thread TCB time
- QWACBSRB for begin ASCB SRB time
- QWACESRB for end ASCB SRB time.

Figure 72 on page 140 shows an example of a DB2 accounting record formatted by DB2PM. You can find a description of the contents of the DB2 accounting record in *IBM DATABASE 2 Administration Guide*. You can also find the description of accounting record fields in member DSNWMSGS, which is shipped in DSN230.DSNSAMP or DSN310.SDSNSAMP.

Figure 80 on page 159 shows the processor time reported for CLASS 1, which is the sum of the times 3, 6, and 9. This is the TCB time measured in the type 101 record. It represents only the work done under the TCB of the DB2 thread. DB2 accounting records are produced when a thread is terminated or a sign-on occurs. This means that the period reported in the DB2 accounting record is the time between thread start or user sign-on (if reusing a thread previously used by another user) and thread termination or another sign-on. With CICS/ESA Version 4 you can use the TXIDSO parameter in the RCT to cause a DB2 accounting record to be produced when the transaction ID changes, as well as when the thread terminates or another sign-on occurs. The period does not contain processor time spent in application code or processor time for creating or terminating the thread.

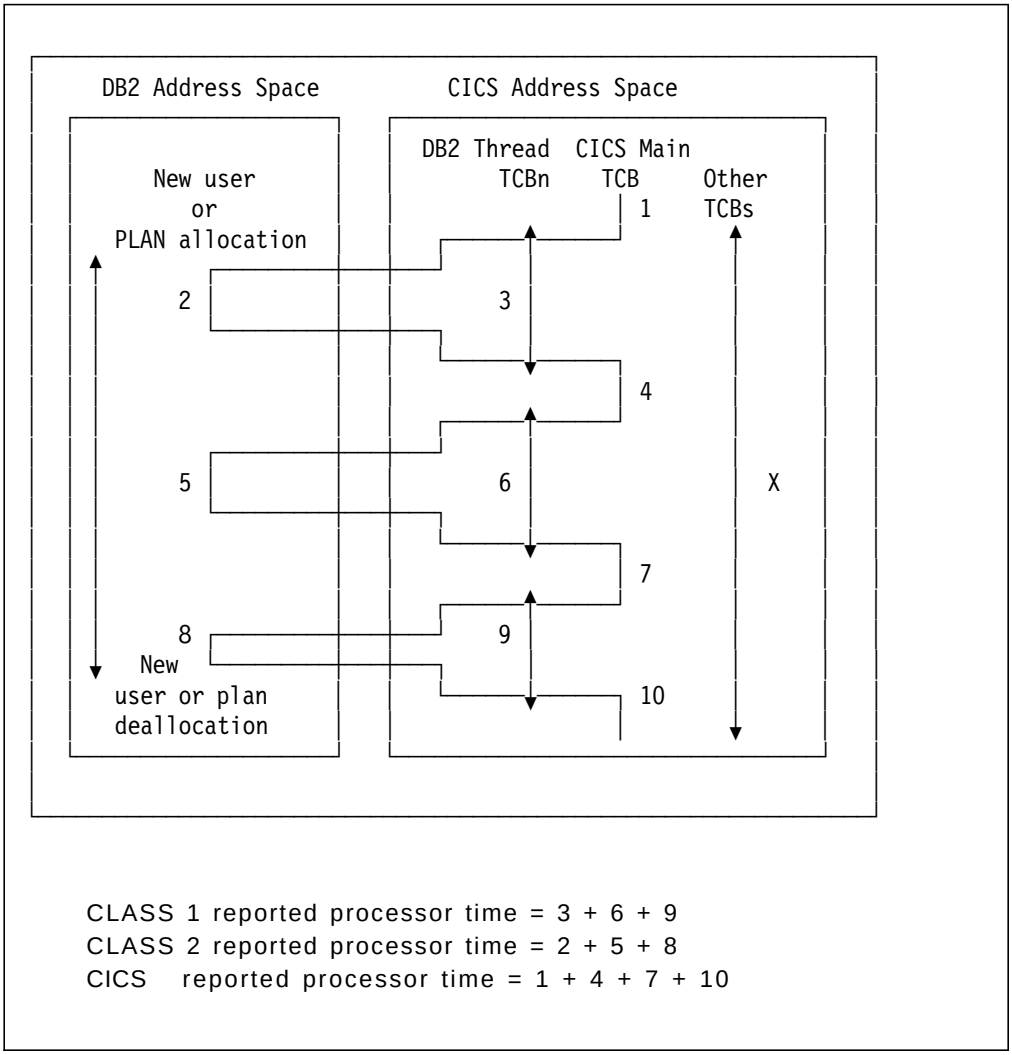


Figure 80. Processor Times Recorded in Type 101 Records for CICS

For thread reuse, this means that many users are included in the same record, which can cause difficulties for both accounting and problem determination.

The TOKENE=YES and TOKENI=YES options in the RCT, together with the new TXIDSO=NO option in CICS/ESA Version 4, provides more granularity, suppresses some strange records, and allows more accurate processor time to be charged to a transaction. This is because a record is produced for each user and involves the passing of a token between CICS and DB2, which is present in both CICS and DB2 traces.

The processor time reported in the DB2 accounting record is not included in the CICS performance records.

For terminal-oriented transactions that issue multiple syncpoints (commit or roll back), the DB2 thread associated with the transaction is released after each syncpoint. (POOL threads are generally terminated, while ENTRY threads can be reused.) If the transaction used a protected thread, this thread can subsequently be reused by either the same transaction, or another transaction that is capable of using the thread. If the transaction used an unprotected thread, it is likely that this thread will be terminated before the first SQL call is issued after the syncpoint, because an unprotected thread is terminated immediately when it is

unused. A new thread must then be created for the transaction. A single transaction can therefore be associated with multiple DB2 threads during the life of its execution.

Because an accounting record gets written at thread termination or at sign-on for a different user, a single transaction (issuing multiple syncpoints and terminal-oriented) can have multiple accounting records associated with it. The processor time is accurately obtainable by authorization ID within the transaction by aggregating all the accounting records for this authorization ID and transaction. Note, however, that the elapsed times associated with the accounting record would be of little value in this case, because the sum of the elapsed times does not include the time from one commit point to the first SQL call in the next LUW. The elapsed time for a thread is associated with the thread and not with the transaction.

A non-terminal-oriented transaction does not release the thread when committing. That means that only one DB2 accounting record is produced for this kind of transaction. However, if other transactions are specified for the same RCT entry, the DB2 accounting record for this thread can include data for these transactions as well, if the thread is reused.

8.3.2.2 Accounting CLASS 2

For accounting CLASS 2, the timer is checked on every entry and exit from DB2 to record the 'IN DB2' time in the SMF type 101 record. In this case, it is the difference that is stored in the record. The fields in the type 101 record used for accounting CLASS 2 processor time are QWACAJST for CICS attach TCB time and QWACASRB for CICS ASCB SRB time.

For a description of the contents of the DB2 statistics records, see *IBM DATABASE 2 Administration Guide*.

The elapsed time (start and end times between the above defined points) is also reported in the type 101 record (QWACASC).

The processor time reported for CLASS 2 is shown in Figure 80 on page 159, and is the sum of the times 2, 5, and 8. Note that in the case of CICS (in contrast to IMS/ESA and TSO), this is not significantly different from the CLASS 1 trace information.

Terminal-driven transactions issuing multiple syncpoints also result in multiple accounting records being written for accounting CLASS 2. However, like accounting CLASS 1, the processor times accurately reflect the processor times 'IN DB2'. Unlike accounting CLASS 1, the CLASS 2 elapsed times accurately reflect the elapsed times spent 'IN DB2'. Accounting CLASS 2 information is very useful for monitoring the performance of SQL execution. However, the processor overhead associated with having class accounting is quite considerable. DB2 Version 3 addresses this problem, and the processor overhead is reduced significantly due to the changing of the IFCID functions invoked.

To reduce the overhead, it was usually recommended that the CLASS 2 trace be limited to trace data for specific authorization IDs, plans and locations. The processor overhead was reduced from 0-15% to 0-5%, with 2% being typical for DB2 Version 3.

8.3.2.3 Processor Times Not Included in Type 101 Record for CICS

Note that the type 101 record for CICS does *not* include:

- Segment 1: Code executed prior to plan allocation, as well as most of the create thread time
- Segments 4 and 7: Time spent executing the application code in the transaction, excluding DB2 calls
- Segment 10: Most of the thread terminate time
- Segment X: Any concurrent work done under other TCBs of the CICS address space, such as journaling and asynchronous VSAM.

8.3.3 Adding CICS and DB2 Transaction Processor Times

If you estimate the total processor time for a single transaction, you could add information from the corresponding CICS performance record and DB2 accounting record. You should take the CPU field from the CICS performance class record. Using the DB2 accounting record, you can calculate the DB2 thread processor time (T1) as:

```
T1 = QWACEJST - QWACBJST
```

This calculates the CLASS 1 TCB processor time. The sum of the processor time from the CICS CPU field and the calculated value of T1 is an expression for the processor time spent in CICS and DB2.

Notes:

- The CLASS 2 processor time is part of the CLASS 1 processor time and should not be added to the sum.
- The processor time used in the DB2 address spaces and recorded in the DB2 statistics records is not related to any specific thread. It can be distributed proportionally to the CPU time recorded in the DB2 accounting records.
- Processor time used in the CICS address space under the subtask TCBs cannot easily be distributed to the CICS performance records, because it includes the processor times for the DB2 subtasks, which are already contained in the calculated T1 value. It means that processor time used in subtasks other than the thread subtasks is not included in the addition.
- Most of the processor time used in the thread TCB to create the thread is not included in any DB2 accounting records associated with this thread, because this processor time is spent before the thread is created.
- The capture ratio for CICS and DB2 should be taken into account.

However, as described later in this chapter, you are not always able to get both a CICS performance record and a DB2 accounting record for a single transaction.

8.4 Accounting Considerations for DB2

You have to perform two different activities when planning your accounting strategy. First, decide the type of data to use in your accounting process (processor usage, I/O, calls, and so on). This involves data that would be meaningful as a basis for accounting. A number of possibilities are given in the following sections.

Second, have this data available at a level that makes it useful for the desired accounting level. Section 8.4.2, “Availability of Accounting Data” on page 164 discusses under which conditions this data is available (the second activity).

8.4.1 Possible Types of Accounting Data

The following data types, which could be used for accounting, are discussed in subsequent sections:

- Processor usage
- I/O
- GETPAGES
- SQL call activity
- Transaction occurrence
- Storage.

8.4.1.1 Processor Usage

The processor usage information given in the DB2 accounting record show in most cases the greater part of the total processor time used for the SQL calls. The DB2 statistics records report processor time used in the DB2 address spaces that could not be related directly to the individual threads.

You should consider distributing the processor time reported in the DB2 statistics records proportionally between all users of the DB2 subsystem (transactions, batch programs, TSO users).

The amount of processor time reported in the DB2 accounting records is (for the same work) relatively repeatable over time.

See 8.3, “Accounting for Processor Usage” on page 155 for more detail on reporting processor usage in a CICS-DB2 environment.

8.4.1.2 I/O

In a DB2 system, the I/O can be categorized in these types:

- Synchronous read I/O
- Sequential prefetch (asynchronous reads)
- Asynchronous writes
- EDM pool reads (DBDs and plan segments)
- Log I/O (mainly writes).

Of these five I/O types, only the synchronous read I/O is recorded in the DB2 accounting record.

The number of sequential prefetch read requests is also reported, but the number of read requests is not equal to the number of I/O.

None of the I/O types should be considered as repeatable over time. They all depend on the buffer sizes and the workload activity.

DB2 is not aware of any caches being used. That means that DB2 reports an I/O occurrence, even if the cache buffer satisfies the request.

8.4.1.3 GETPAGES

GETPAGE represents a number in the DB2 accounting record that is fairly constant over time for the same transaction. It shows the number of times DB2 requested a page from the buffer manager. Each time DB2 has to read or write data in a page, the page must be available, and at least one GETPAGE is counted for the page. This is true for both index and data pages. How often the GETPAGE counter is incremented for a given page used several times depends on the access path selected. However, for the same transaction accessing the same data, the number of GETPAGEs remains fairly constant over time, but the GETPAGE algorithm can change between different releases of DB2.

If the buffer pool contains the page requested, no I/O occurs. If the page is not present in the buffer, the buffer manager requests the page from the media manager, and I/O occurs.

The GETPAGE number is thus an indicator of the activity in DB2 necessary for executing the SQL requests.

8.4.1.4 Write Intents

The number of set write intents is present in the QBACSWS field of the DB2 accounting record, but the number is not related to the actual number of write I/Os from the buffer pools. The number represents the number of times a page has been marked for update. Even in a read-only transaction this number can be present, because the intended writes to the temporary work files used in a DB2 sort are also counted.

The typical case is that the number of set write intents is much higher than the number of write I/Os. The ratio between these two numbers depends on the size of the buffer pool and the workload. It is not a good measurement for write I/O activity, but does indicate the complexity of the transactions.

8.4.1.5 SQL Call Activity

The number and type of SQL calls executed in a transaction are reported in the DB2 accounting record. The values are repeatable over time, unless there are many different paths possible through a complex program, or the access path changes. The access path chosen can change over time (for example by adding an index).

A given SQL call can be simple or complex, depending on factors such as the access path chosen and the number of tables and rows involved in the requests.

The number of GETPAGEs is in most cases a more precise indicator of DB2 activity than the number of different SQL calls.

8.4.1.6 Transaction Occurrence

A straightforward way of accounting is to track the number and type of transactions executed. Your accounting is then based on these values.

8.4.1.7 Storage

The DB2 accounting record does not contain any information about real or virtual storage related to the execution of the transactions. One of the purposes of the DB2 subsystem is to optimize the storage use. This optimization is done at the DB2 level, not at the transaction level.

A transaction uses storage from several places when requesting DB2 services. The most important places are the thread, the EDM pool, and the buffer pools.

Because no information is given in the DB2 accounting record about the storage consumption and because the storage use is optimized at the subsystem level, it is difficult to account for storage in a DB2 environment.

8.4.2 Availability of Accounting Data

When you decide which resources to account for, you must find method to supply the data you need. This section assumes that you need the DB2 accounting records to give this information. In the next sections we give two methods for relating the DB2 accounting records to the end user. They are:

- Using the DB2 authorization ID
- Combining CICS and DB2 records.

8.4.2.1 Using the DB2 Authorization ID

One possibility is to give each end user a different authorization ID from a DB2 viewpoint. This can be done by specifying the RCT parameter AUTH as USER, USERID, GROUP, or TERM. A DB2 accounting record is generated that contains data only for this authorization ID. Accumulating all the DB2 accounting records by authorization ID provides a basis for accounting.

Note that the resources spent in CICS and DB2 can be accumulated independently. Matching the CICS performance records and the DB2 accounting records is not necessary. However, this method also has disadvantages. These are discussed in the section 4.4.3, “Plan Execution Authorization” on page 69, but are repeated here for convenience. The disadvantages for large networks are:

- Many GRANT statements are required.
- Many entries are in SYSPLANAUTH and SYSPACKAUT.
- Maintenance can be complicated, because there can be many combinations of authorization ID (USER, USERID, GROUP or TERM) and plans.
- The overhead involved in making authorization checks for many transactions (new authorization ID at thread reuse is likely).

If dynamic SQL is being used in the CICS applications, the authorization ID also needs the privileges required to access the DB2 resources involved.

From a usability and performance viewpoint, specifying USER, USERID, GROUP and TERM is not an attractive solution.

8.4.2.2 Combining CICS and DB2 Records

If your accounting is based on the actual data from the DB2 accounting records, and you did not choose the method of using the DB2 authorization ID, you need to combine the DB2 accounting records with the CICS performance records. By doing so, the DB2 resources from the DB2 accounting records are related to the CICS performance records, and these records can be related to the end user.

Other reasons for trying to combine the two-record types can exist. If the purpose of your accounting is not to charge the end-user, but to analyze the resources used by the individual CICS transactions, you need to combine these records.

If you use either the RCT parameter TOKENE or TOKENI, CICS passes its LU 6.2 token to DB2 to be included in the DB2 trace records. The token is written to QWHCTOKN in the correlation header. Using TOKENE or TOKENI can simplify the task of correlating CICS and DB2 accounting records. If you do not use TOKENE or TOKENI, there is no easy way to do this matching. Two problems exist:

- There is not a one-to-one relationship between the CICS performance records and the DB2 accounting records. A DB2 accounting record can contain information about one CICS transaction, multiple CICS transactions, or part of a CICS transaction.
- No fields in the DB2 accounting record can be used to exactly identify the corresponding CICS performance record. This is because there is no one-to-one relationship between the two record types.

Below, we show what you can do when you are not using TOKENE, TOKENI, or TXIDSO (CICS/ESA Version 4 only) to correlate CICS and DB2 accounting records.

Fields in the DB2 accounting record that can be used to correlate to the CICS performance records are:

- Correlation ID, containing the CICS 4-character transaction code
- The timestamp fields[5]
- The authorization ID field.
- The CICS LU 6.2 token.

Note that there is no ideal way of combining the two record types. There will be cases where it is impossible to make the matching correct because the transactions are run concurrently.

Figure 81 on page 166 shows four different accounting cases.

[5] Note that the time stamps used in CICS and DB2 are based on different times. The CICS timer values are based on the MVS TOD clock timer. The DB2 timer values are based on the hardware Store Clock timer.

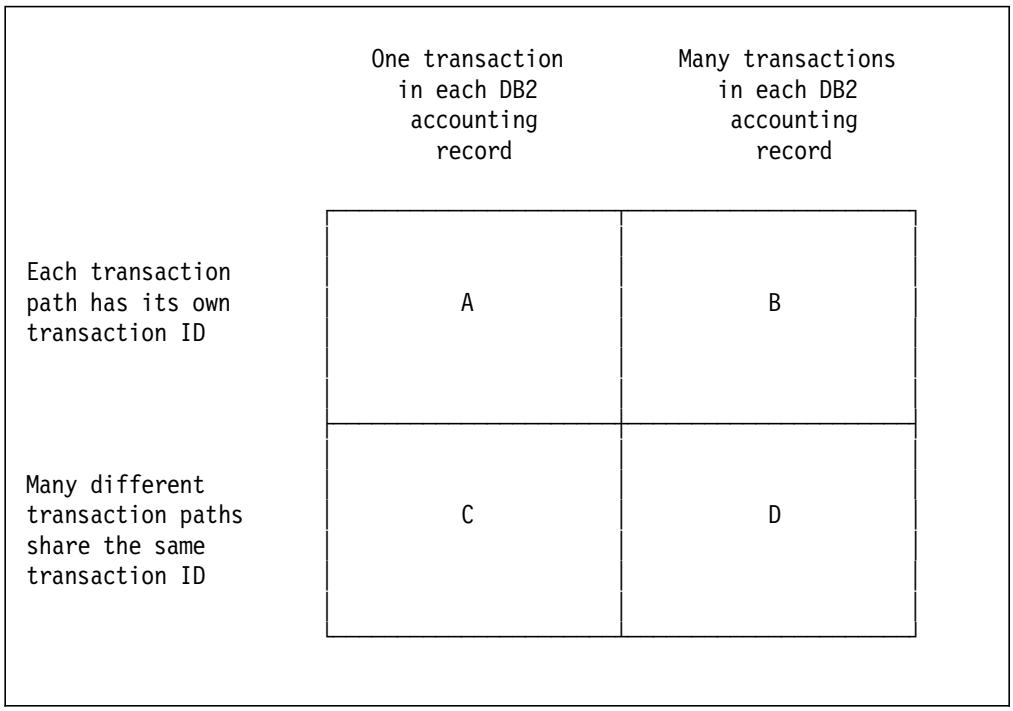


Figure 81. Different Accounting Cases. Four combinations of using transaction codes and the number of transactions reported in one DB2 accounting record.

The CICS application can be designed so that the work done under a specific CICS transaction ID is the same between executions. Cases A and B describe this situation.

In cases C and D, many different transaction paths use the same CICS transaction ID. The work done and the resources consumed differ between executions.

A typical example for cases C and D is where the terminal user has a menu displayed at the terminal and can choose different options for the next transaction. If the previous transaction was terminated by setting up the CICS transaction ID with the EXEC CICS RETURN TRANSID(zzzz) command, then the next transaction runs under the transaction ID zzzz, no matter what the terminal user chooses.

For more details about the CICS transaction ID and plan considerations, see section 5.4, “Resource Control Table Parameters” on page 82.

Transactions can also be divided into two groups depending on when the DB2 accounting record is written. In cases A and C, the DB2 accounting record contains information for just one transaction. A DB2 accounting record is always written at thread termination or at the sign-on of a new authorization ID, reusing the same thread. The use of the TOKENE parameter causes a DB2 accounting record to be written at the end of the transaction, even with thread reuse.

Remember that to obtain the situation where each DB2 accounting record contains information for just one transaction, you must avoid thread reuse, unless you used TOKENE, TOKENI, or TXIDSO. In cases B and D, the thread contains information from many transactions.

8.4.2.3 Four Accounting Cases

The four different accounting cases, Case A through Case D, are discussed in more detail below.

Case A There is one DB2 accounting record for each CICS performance record if the transaction made any SQL calls. If the user can combine the two corresponding record types, all information is available for the specific transaction.

The end user can be identified in the CICS performance record. This record contains the CICS activities related to this transaction. The DB2 accounting record can be identified by the CICS transaction code, start and end time for the thread, and authorization ID depending on the value of the AUTH parameter in the RCT.

It is expected that all transactions consume comparable amounts of resources, because all transactions are identical.

Case B In this case it is not possible to relate the DB2 accounting record directly to the CICS performance record, because many transactions use the same thread.

If only one CICS transaction type is using the plan for this thread, then the resources in DB2 can be split equally between each transaction. This is reasonable, because only one transaction type is using this CICS transaction code. The transactions are (by definition) almost identical.

The number of commits and backouts in the DB2 accounting record indicates the number of transactions covered in this record. However, information about a terminal-oriented transaction issuing multiple commits can be contained in more than one DB2 accounting record, because it releases the thread for each commit.

If two or more different transactions use the same RCT entry, the method of distributing the resources equally may not be used, because the different transactions can use different resources.

In that case another technique can be used. The user can periodically measure the accurate resources used by each transaction type. This can be done by not allowing thread reuse. The output from these measurements can then be used as model transactions.

The DB2 accounting records are then not used directly in the accounting process, but can be used to validate the correctness of the model transactions. Measured on a daily basis for example, the weight of the model transactions multiplied with the corresponding number of transactions from the CICS performance records should equal the accumulated numbers from the DB2 accounting records.

Note that a transaction reusing a thread is not using any processor time for starting the thread.

Case C This situation is similar to case A, except that the number of transactions for a specific user is not a good measurement of the resources consumed. All individual CICS performance records and the corresponding DB2 accounting records should be accumulated for each user, because many transactions use the same CICS transaction code.

The principle for correlating the two records types is the same as in case A.

Case D In this case one DB2 accounting record covers many different CICS transactions. Model transactions can be defined, but it is difficult to decide which one to use for a specific CICS transaction, because the same CICS transaction code is used by transactions running through different program paths.

In some situations other fields in the CICS performance record can be used to distinguish one transaction from another. For example, the user could supply information in a user field in the CICS performance record. This information could define the transaction being executed. The model transactions can then be used.

8.5 Summary of Accounting for DB2 Resources

Accounting can be a complicated process in a CICS-DB2 environment. Generally, CICS resources are recorded in CICS statistics and performance records, and DB2 resources are recorded in DB2 accounting and statistics records.

One of the main decisions in a CICS/DB2 environment is whether to base the accounting on information from the individual transactions executed or on some kind of averaging method.

- If the resources used in the individual transaction is the basis for accounting, both the CICS performance records and the DB2 accounting records can be related to the specific end user. Two methods are available to do this:
  - Running all DB2 activity under the authorization ID of the end user. This is obtained by setting the AUTH parameter in the RCT to either USER, USERID, or TERM, whichever best describes the end user in the given situation.

    In this case there is no need to combine the CICS performance records and the DB2 accounting records. The CICS and the DB2 resources consumed in the transaction are accumulated separately for each user.

    The disadvantages of this method are performance and maintenance aspects, as shown in the previous sections.
  - Combining the DB2 accounting records with the CICS performance records. The combination can only be done approximately, because there is no one-to-one relationship between the two record types.

    The four different cases described above indicate that if the accounting process in a CICS-DB2 environment must be based on the individual DB2 accounting records, then the overall application design should account for this situation.
- Alternatively you could define and calibrate a number of model transactions, measure these transactions in a controlled environment, and count only the number of model transactions executed by each end user.

  If several transactions use the same CICS transaction ID, you can mark the CICS performance record with an identifier that is unique to each transaction.
- You could use the TOKENE, TOKENI, or TXIDSO parameter on the RCT definition to pass the CICS LU 6.2 token to DB2 for correlation purposes.

This carries out an overhead for each transaction that specifies these parameters. However, they are very useful in those cases where accounting is necessary.

Often you must determine whether performance or accounting has the highest priority. The application design and the RCT tuning options are likely to be different in these two cases.

If the detailed information from the DB2 accounting records is available, the user has many possibilities for defining the cost formula based on the DB2 accounting records.

- Where repeatability combined with a reasonable expression for the complexity of the transactions has high priority, then the processor usage, the GETPAGE count, and the set write intents count are good candidates.
- If the purpose of the accounting process is to analyze the behavior of the CICS transactions, then any information in the DB2 accounting records can be used.

In summary, it is the responsibility of the installation to design the applications and to tune the RCT entries with respect to the trade-off between performance and accounting requirements.

Chapter 9. CICS-DB2 Recovery and Restart

In this chapter we discuss the procedures you need to have in place for recovery and restart in a CICS-DB2 environment.

9.1 Overview

In a CICS-DB2 environment, a transaction can update:

- Resources controlled by CICS, for example VSAM files and recoverable temporary storage queues
- Resources controlled by DB2 (that is, tables)
- Resources controlled by another resource manager, for example IMS/ESA database control (DBCTL), or the IBM Message Queue Manager (MQM) for MVS/ESA.

The recovery of these distributed resources during a restart operation works as follows:

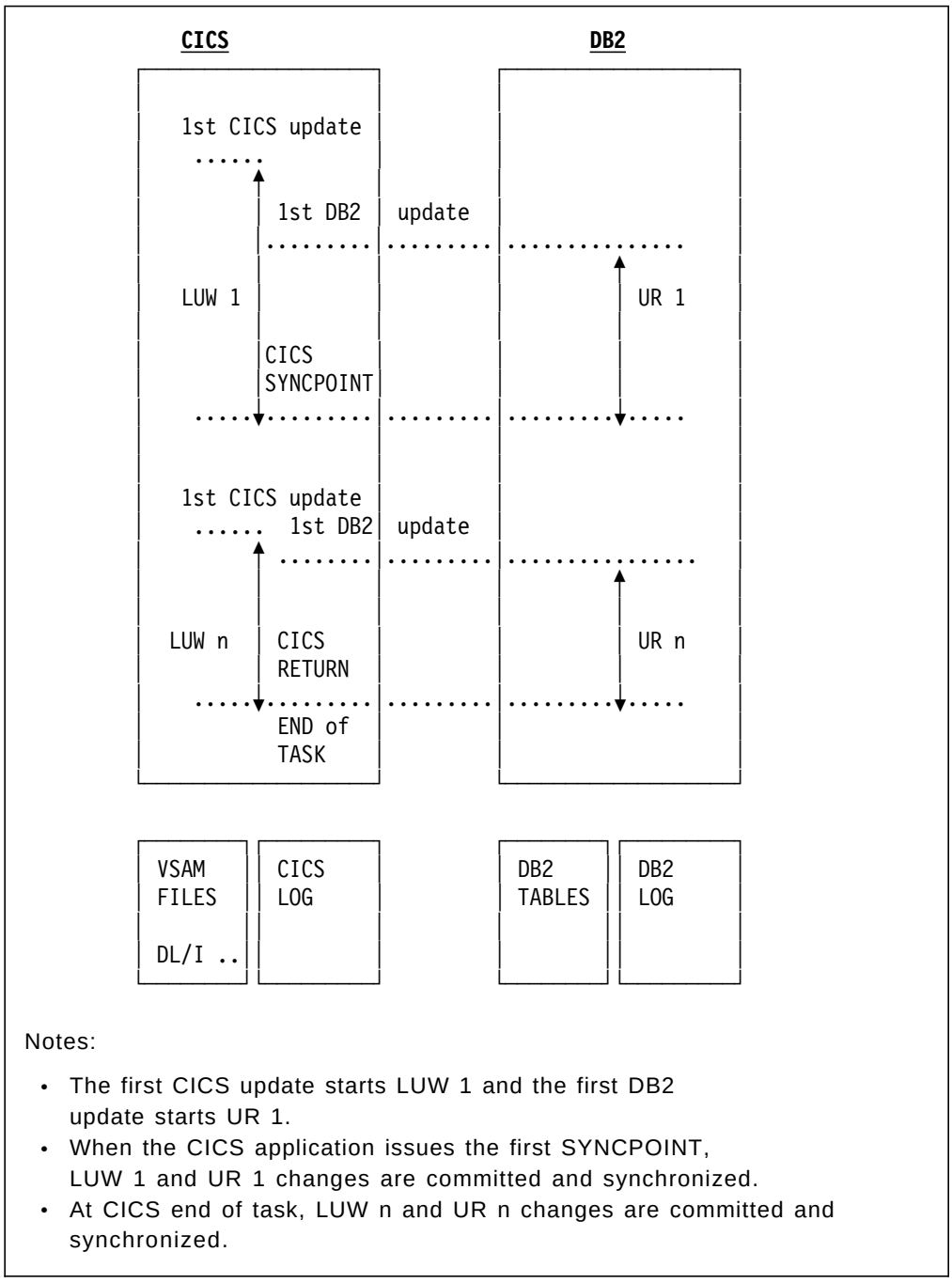
- CICS recovers its resources. There is no difference in the way this is done as compared to a CICS-only environment. The same locking protocol is used and the CICS log contains change information for these resources only. Refer to the *CICS/ESA Recovery and Restart Guide* for information on this process.
- DB2 resources are controlled and recovered by DB2. See the *IBM DATABASE 2 Administration Guide* for details on this recovery.

  DB2 uses its own log and locking protocol. Nevertheless, DB2 needs to coordinate its recovery with that done by CICS, in case of in-doubt threads. This coordination is implemented using the two-phase commit protocol (TPCP), which is explained in the next section of this chapter.

  The CICS attachment facility can also use single-phase commits or read only commits to improve performance, depending on the operation.

CICS uses the term logical unit of work (LUW) in the same sense that DB2 uses the term unit of recovery (UR). For both subsystems, the terms mean the updates performed in either subsystem by a transaction, between two points of consistency.

The start of a CICS task defines a point of consistency and each syncpoint defines a new point of consistency. The term UR refers to DB2 or to a subsystem in general. The term LUW only refers to CICS. Figure 82 on page 172 shows this relationship.



Notes:

- The first CICS update starts LUW 1 and the first DB2 update starts UR 1.
- When the CICS application issues the first SYNCPOINT, LUW 1 and UR 1 changes are committed and synchronized.
- At CICS end of task, LUW n and UR n changes are committed and synchronized.

Figure 82. CICS-DB2 Recovery and Restart

9.2 Two-Phase Commit Protocol

This protocol recognizes the existence of two distinct phases of processing required to establish a new point of consistency (synchronization point). One of the connected subsystems is the coordinator, the others are participants:

- In connections with IMS, CICS, or DL/I batch:
  - IMS, CICS, or DL/I batch are always the coordinators
  - DB2 is always the participant.

- In connections with TSO/batch, DB2 is the coordinator and there is no participant outside of DB2.

9.2.1 Illustration of the Two-Phase Commit

Figure 83 on page 174 illustrates the two-phase commit process. The numbers in the following discussion are keyed to those in the figure.

Point 1 The data in CICS is at a point of consistency.

Point 2 A CICS application program calls DB2 to update some data by executing an SQL statement.

Point 3 This starts a unit of recovery in DB2.

Point 4 Processing continues in CICS until an application synchronization point is reached at Point 4.

Point 5 CICS starts commit processing.

To do that, CICS uses:

- A syncpoint command, or
- A PSB normal termination command, or
- A normal application termination.

Phase 1 of commit processing begins.

Point 6 As CICS begins Phase 1 processing, so does DB2.

Point 7 DB2:

- Successfully completes Phase 1
- Writes this fact in its log
- Notifies the coordinator.

Point 8 CICS receives the notification.

Point 9 CICS successfully completes its Phase 1 processing.

Now both subsystems agree to commit the data changes, because both completed Phase 1, and could recover from any failure.

CICS records in its log the instant of commit, that is, the irrevocable decision of the two subsystems to make the changes.

Point 10 CICS begins Phase 2 of the processing, that is, the actual commitment. It notifies DB2 to begin its Phase 2 processing.

Point 11 DB2 logs the start of Phase 2.

Point 12 DB2:

- Completes its Phase 2 successfully.
- Notifies CICS that it is finished with Phase 2.

This is a new point of consistency for DB2.

Point 13 CICS finishes its Phase 2 processing.

The data controlled by both subsystems is now consistent and available to other applications.

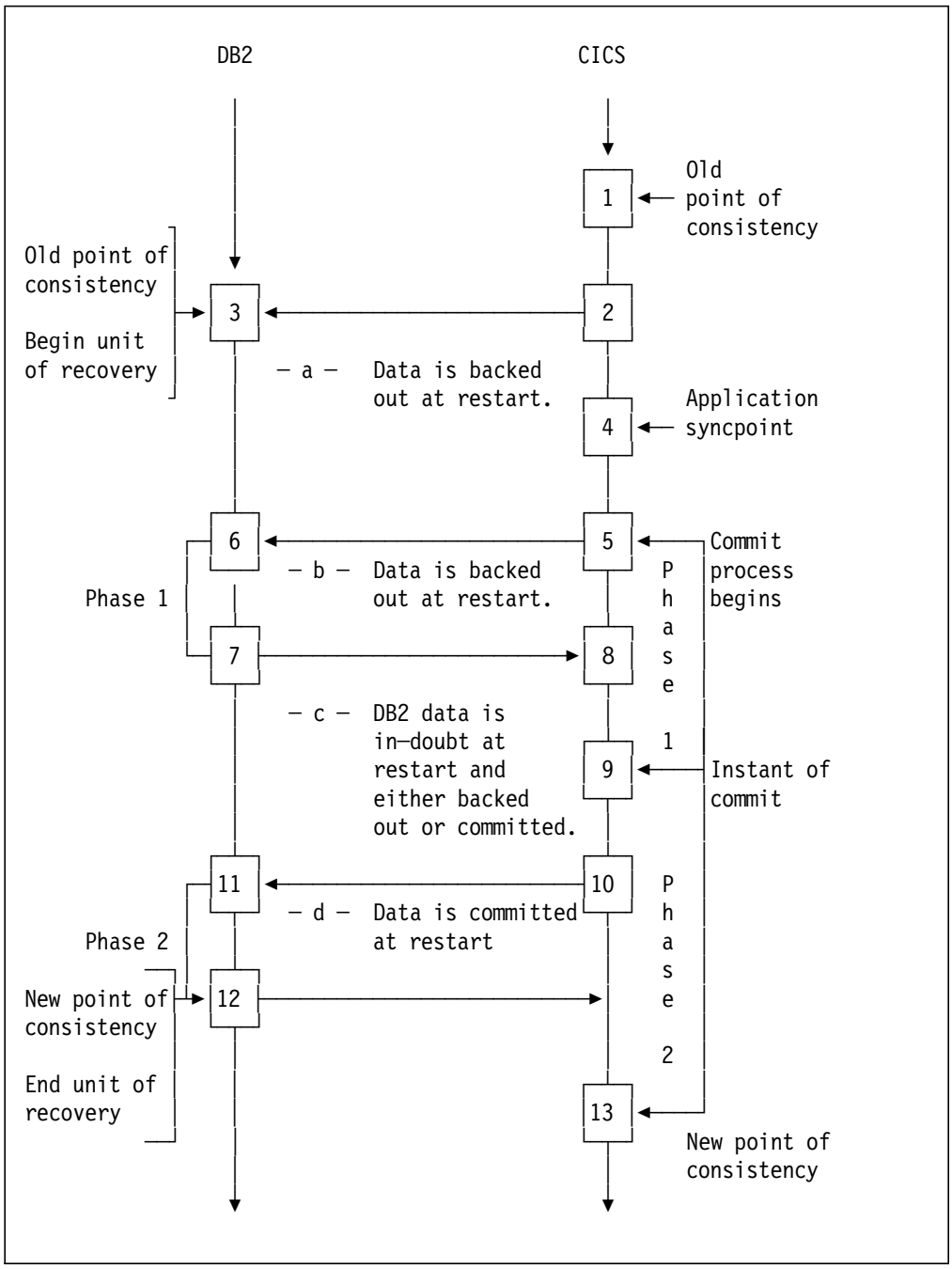


Figure 83. Restart Coordinated between CICS and DB2

9.2.2 Maintaining Consistency after a Failure

If DB2 fails while connected to CICS, it must determine during restart whether to commit or to roll back units of recovery that were active at the failure time. For certain URs, DB2 has enough information to make the decision. For others, it must get information from CICS when the connection is reestablished.

The status of a UR after a failure depends on the moment of the failure. Figure 83 helps to illustrate the possible statuses listed below:

STATUS DESCRIPTION and PROCESSING

Inflight DB2 failed before finishing Phase 1 (period a or b). During restart, DB2 backs out the updates.

In-doubt DB2 failed after finishing Phase 1 and before starting Phase 2 (period c). Only CICS knows if the failure happened before or after the commit (Point 9):

- If it happened before, DB2 must back out its changes.
- If it happened after, DB2 must make its changes and commit them.

At restart, DB2 waits for information from CICS before processing this UR.

In-commit DB2 failed after it began its own Phase 2 processing (period d). It commits the changes.

In-abort DB2 failed after a UR began to roll back, but before the process was complete (not shown in the figure). During restart, DB2 continues to back out the changes.

9.2.3 More on the Two-Phase Commit

Figure 84 on page 177 shows the two-phase commit protocol between CICS and DB2 in the case of a positive DB2 vote.

Phase 1 Commit preparation and vote collection.

The coordinator (CICS) assumes the role of the vote collector and requests that all participants (only DB2 in this case) prepare to commit.

Each participant must declare whether or not it agrees to continue with the commit process.

- If one participant disagrees, it returns a negative vote and the unit of recovery must be rolled back by all the subsystems involved.
- Each participant agreeing to commit must prepare to do the second phase (that is, complete its logging during Phase 1), but still be capable of reversing the changes if a negative vote is cast by some other participant.

In the CICS case, when the CICS application program reaches a syncpoint, CICS asks DB2 to prepare to commit changes made to DB2 resources by this transaction since the last syncpoint. DB2 then logs the updates but retains all locks.

When the logging is successfully completed, DB2 returns a positive response to CICS. If DB2 is unable to commit the updates, it returns a negative response, and the updates to the CICS and DB2 resources are rolled back. A positive response from DB2 indicates that it is able to commit or reverse the changes if required.

CICS then logs any pending updates made to its resources, but retains all locks.

Phase 2 Must-complete processing.

The coordinator, having received all positive votes from phase 1, notifies the participants to start Phase 2, sending a commit. Each participant then commits the unit of recovery and makes the new committed form of the objects (that is, database records) accessible to other applications. There are no voting options in Phase 2.

If any vote from Phase 1 was negative, the coordinator notifies all participants to roll back the changes.

If any subsystem abends after Phase 2 commit is started, that subsystem must be able to recover to the synchronization point with the unit of recovery completed (that is, the commit Phase 2 must be completed during subsystem restart).

In the CICS case, when the CICS Phase 1 logging is successfully completed, CICS requests DB2 to commit. This begins Phase 2.

DB2 then commits its changes and releases all locks.

9.2.4 Log Protocol

An essential part of the two-phase commit protocol (TPCP) is the log protocol. Figure 85 on page 179 shows CICS and DB2 logs related by the control flow of the TPCP. Note the following log records ordered by the time they are written to each log:

- The *Begin UR DB2 log record* signals the beginning of a UR in DB2. There is no equivalent log record in CICS. The first CICS log record for a transaction contains a start-of-task flag that plays a similar role.
- The *End Phase 1 DB2 log record* signals the end of Phase 1 in DB2. This record is forced to the log and so flushes any pending change log records for this transaction. Since DB2 is a participant, this record does not mean (as in the CICS case) that the updates are committed. It means that they can be committed or backed out according to the CICS Phase 2 direction.

  Note that the record is in the log before sending the OK to the PREPARE.

  In the case of a failure before writing this record, the UR is backed out.
- The *CICS Physical End Of Task log record* signals the end of the task (EOT), plus the Phase 1 to Phase 2 transition (End of Phase 1-Start of Phase 2).

  When this record is logged, all CICS resources are committed. This record is forced to the log, so the CICS task does not regain control until it is externalized. Note that any other log records for the transaction that might still be in the buffer are flushed at this time. This record also means that DB2 voted yes to Phase 1, and so this phase is considered complete.

  If a failure occurs before writing this record, CICS backs out the LUW.

  Once this record is externalized, the Phase 2 COMMIT is broadcast.
- The *Begin Phase 2 DB2 log record* is also forced and records that the final direction for the commit protocol was received from CICS.

  A DB2 UR is in-doubt from the time the End Phase 1 record is written until the time the Begin Phase 2 record is written. If CICS, DB2, or the CICS attachment facility fails during this time interval, the UR remains in-doubt until both systems are running again and reconnected.

  When the Begin Phase 2 record is logged, the *UR* is committed.

  There is no in-doubt window for CICS because being the coordinator results in only one log record signaling the Phase 1 to Phase 2 transition.
- As the answers to the CICS Phase 2 COMMIT are received, an *RMI Forget* CICS log record is written to the CICS log buffer for each participant but not forced to the log. This record represents that the participant received and acknowledged the Phase 2 COMMIT.

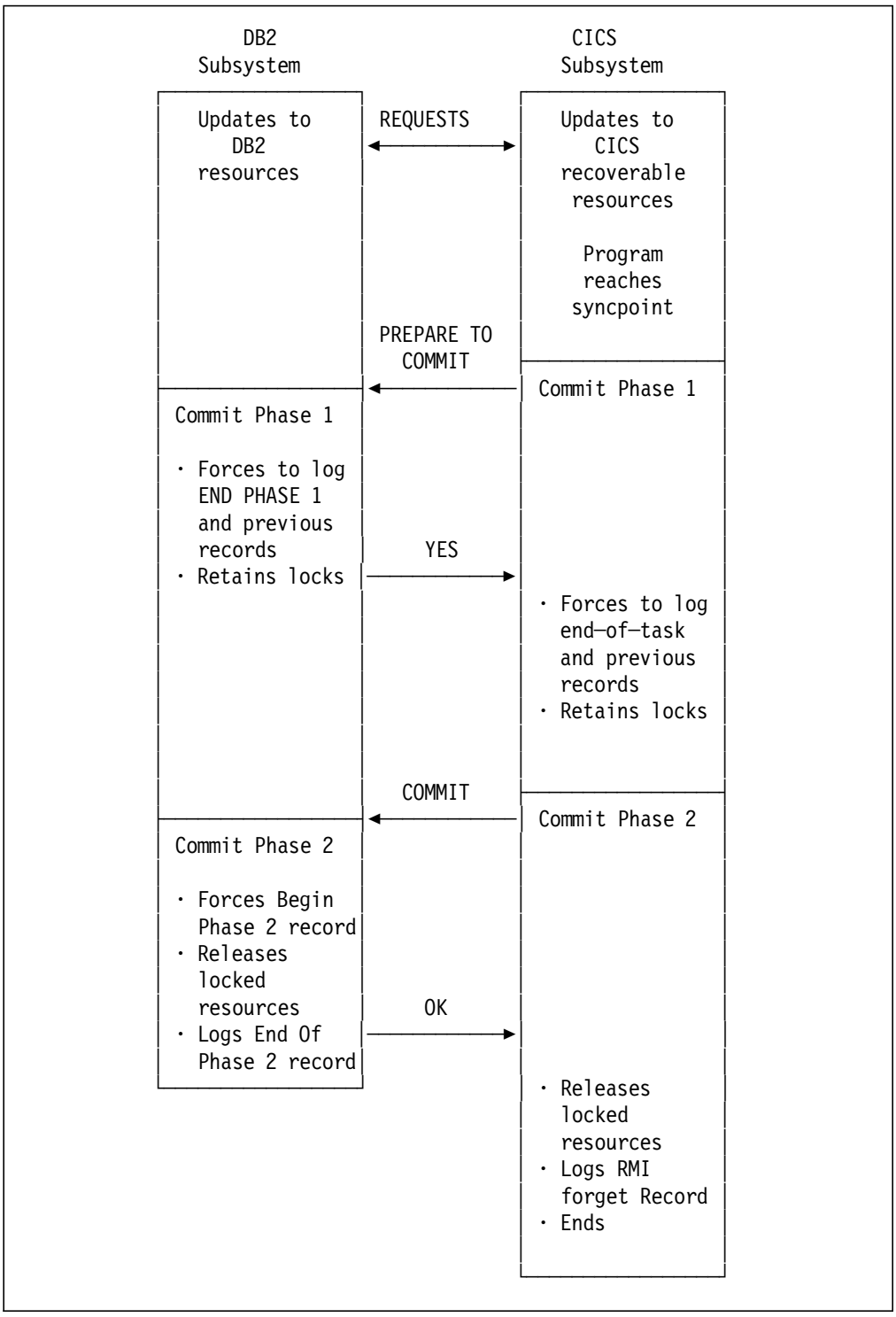


Figure 84. Diagram of the CICS-DB2 Two-Phase Commit Protocol

When the RMI Forget record is in the log, CICS forgets about the DB2 UR, it is no longer in-doubt. The record has no meaning for CICS resources.

- The *End Phase 2 log record* means that DB2 ended any process related to this UR. This record is not forced to the log.

9.2.5 CICS-and DB2-Related Log Records

Figure 85 on page 179 shows the CICS and DB2 log records.

The DB2 log records presented in Figure 85 on page 179 are:

- Begin UR, which signals the start of the DB2 UR.
- The necessary UNDO/REDO records for the DB2 changes.
- The DB2 log records related to syncpoint (End Phase 1, Begin Phase 2 and End Phase 2).

Because no CICS resources were updated in this transaction, only log records related to the commit process appear:

- Logical Start of Syncpoint
- RMI Prepare
- Physical End of Task
- RMI Forget.

DB2 provides a service aid, called DSN1LOGP, which reads the contents of the DB2 log and formats the contents for display. Two report formats are available:

- A detailed report formats and displays individual log records. This information is designed to help IBM support center personnel resolve problems that require extensive analysis of the log contents.
- A summary report summarizes the information contained in the log to help you perform a conditional restart. Since DSN1LOGP allows you to specify optional range criteria, such as the name of one or more units of recovery (URIDs), or the name of a single database, this summary report for the DB2 log can be used in correlation with the CICS log information.

For details on how to use the DSN1LOGP service aid, see *IBM DATABASE 2 Diagnosis Guide and Reference* for your release of DB2.

9.3 Other Commit Protocols

There are also two other commit processing protocols in the CICS attachment facility:

- *Single-phase commits*. These occur when DB2 is the only resource manager that CICS is communicating with in this unit of recovery (UR). CICS does no logging at all when DB2 is the only resource manager holding any updates. If other resource managers (including CICS itself) are holding any updates as well as DB2, then CICS drops into two-phase commit. Figure 86 on page 180 diagrams the flow of a single-phase commit and shows the DB2 log records.
- *Read-only commits*. These occur when there is another updater besides DB2, but the DB2 access was read-only. There is no DB2 logging for read-only commits, because DB2 does not start logging a unit of recovery until recoverable work is done. Figure 87 on page 181 diagrams the flow of a read-only commit and shows the DB2 and CICS log records.

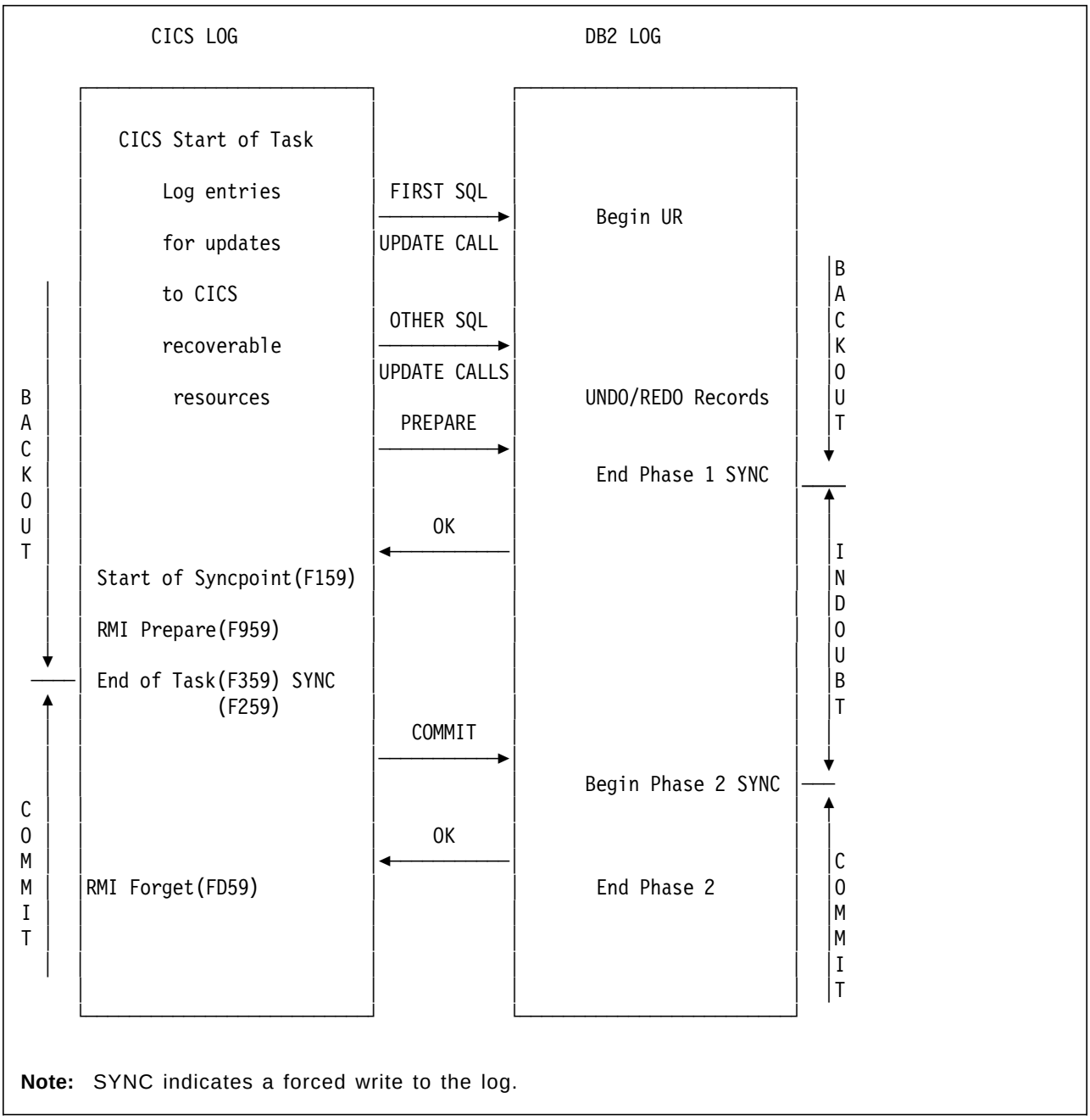


Figure 85. CICS and DB2 Log Records for a CICS Task

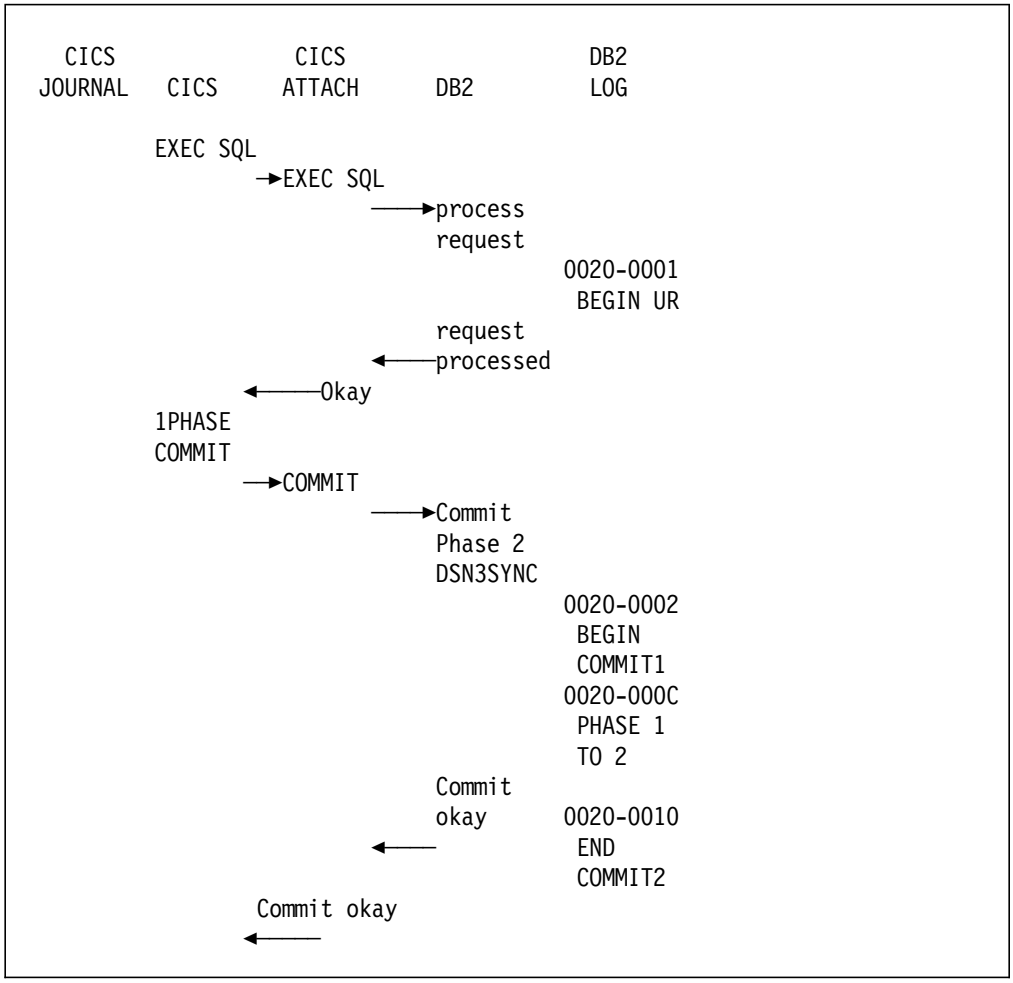


Figure 86. CICS-DB2 Single-Phase Commit

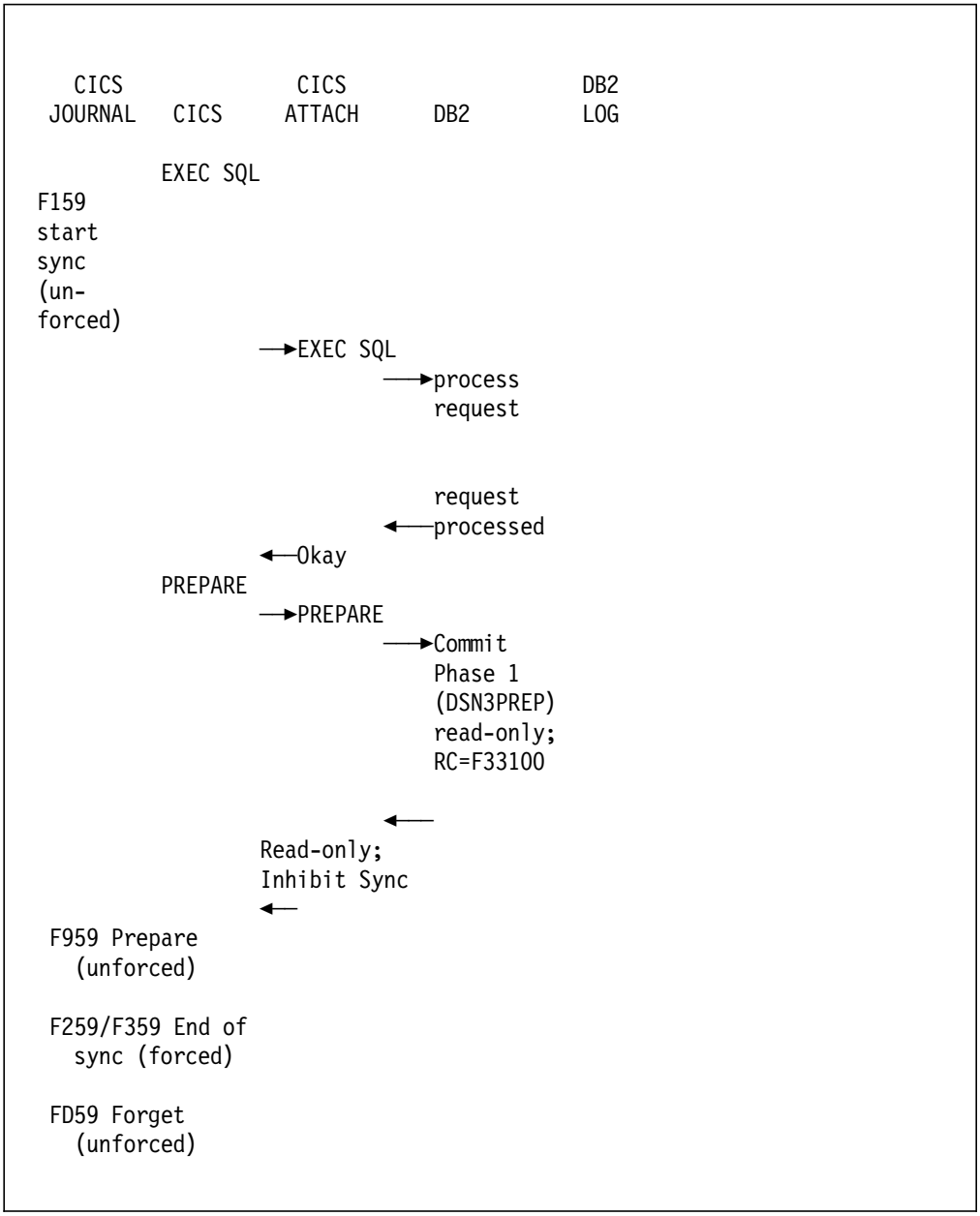


Figure 87. CICS-DB2 Read-Only Commit

9.3.1 CICS Attachment Facility Commit Statistics

You cannot prevent collection of statistics. You display the current statistics by:

- Issuing the DSNC DISP STAT command, or
- Stopping the CICS attachment facility, which produces the statistics automatically. The shutdown statistics report is sent to the RCT SHDDEST= destination.

See Figure 67 on page 129 for more information on statistics and the DSNC DISP STAT command.

9.4 The Restart Process after Failure

Possible types of failures and their associated restart procedures are:

- If both DB2 and CICS fail (for example, because of a hardware or operating system failure), both systems must be restarted and the connection reestablished.
- When either DB2 or CICS fails, the failing subsystem must be restarted and the connection reestablished.
- If the CICS-DB2 connection fails, the connection must be reestablished.
- A CICS transaction abend is automatically backed out on both susbsystems. The transaction must be resubmitted when the failure cause is corrected.

The following sections discuss each of the preceding restart procedures. Remember that messages for the CICS attachment facility shipped with CICS/ESA Version 4 are of the format DSN2xxxx instead of DSNCxxxx.

9.4.1 CICS Restart

CICS restart performs two actions:

- Backs out any inflight LUW
- Builds a resolution list of the possible DB2 in-doubt URs, indicating for each of these the CICS final direction (back out or commit).

Both actions are based on information in the CICS log:

- An LUW is committed or backed out, based on whether the corresponding physical or logical end-of-task record is in the CICS log or not.
- A DB2 UR can be in-doubt if the RMI forget record is not in the CICS log.

When the connection is reestablished, the resolution list is used to solve DB2 in-doubt URs. CICS resources are not affected by this resolution process. CICS is the coordinator and decides to commit or back out, based on information in its own log.

9.4.2 DB2 Restart

Three different situations can arise in DB2 URs following an abnormal termination. These are:

- The End Phase 1 entry is missing in the log.

  DB2 backs out changes made to its resources in this instance. CICS does the same because it did not receive a positive response to its PREPARE TO COMMIT request.

- The End Phase 1 is present, but the Begin Phase 2 is missing.

  In this case the UR is in-doubt because the action taken by CICS for this unit of work is unknown.

  DB2 commits or abends the changes to its resources after the connection is reestablished and CICS indicates what action to take.

  The next section describes connection reestablishment.

- The Begin Phase 2 entry is present, but the End Phase 2 entry is missing.

DB2 commits the changes in this instance because it knows that CICS already committed any changes to its resources.

9.4.3 Reestablishing the CICS-DB2 Connection

Abnormal termination of the connection between CICS and DB2 result in in-doubt URs. DB2 cannot resolve these in-doubt URs (that is, commit or abend the changes made to DB2 resources) until the connection to CICS is restarted.

This means that CICS should always be automatically restarted (START=AUTO specified in the SIT) to get all necessary information available for in-doubt thread resolution from its log.

Where the CICS attachment facility is abnormally terminated, CICS and DB2 build in-doubt lists either dynamically or during restart, depending on the failing subsystem.

For CICS, a DB2 UR can be in-doubt if the RMI Forget entry (FD59) is absent. Note that the in-doubt applies only to the DB2 UR, because CICS already committed or backed out any changes to its resources.

A DB2 UR is in-doubt for DB2 if an End Phase 1 is present and the Begin Phase 2 is absent.

A process to resolve in-doubt URs is initiated during startup of the CICS attachment facility. During this process:

- The CICS attachment facility receives a list of in-doubt URs for this connection ID from DB2 and passes them to CICS for resolution.
- CICS compares entries from this list with entries in its own. CICS determines from its own list what action it took for the in-doubt UR.
- For each entry in the list, CICS creates a task that drives the CICS attachment facility, specifying the final commit or abort direction for that UR.
- If DB2 does not have any in-doubt URs, a dummy list is passed. CICS then purges any unresolved URs from previous connections.

If CICS cannot resolve any of the in-doubt URs, or finds inconsistencies during this process, it informs the CICS attachment facility. The CICS attachment facility then sends either a DSNC001I, DSNC034I, DSNC035I, or DSNC036I message to the starting terminal to inform installation personnel of this condition. This situation can arise when:

- CICS is cold-started and therefore cannot build an in-doubt list (DSNC001I).
- A UR is not in-doubt for DB2, but must be according to CICS log information (DSNC034I). Probably it is a software error. The in-doubt UR can also be caused by a manual resolution of an UR after it is presented to CICS for resolution, but before it is resolved.
- A UR is in-doubt for DB2, but must not be according to CICS log information (DSNC035I). The in-doubt UR can be caused by CICS not using the correct log.
- DSNC036I always represents a software error.

See section 9.4.4, “Manual Resolution of In-doubt URs” on page 184 for instructions about these in-doubt URs that CICS cannot resolve. *Manual resolution is an exception procedure*. CICS retains details of in-doubt URs that

were not resolved during connection startup. An entry is purged when it no longer appears on the list presented by DB2, or when present, DB2 solves it.

9.4.3.1 Removed Recovery Restriction

CICS/ESA Version 4 improves the coordination between CICS and DB2 during recovery in an environment where a CICS region is capable of connecting to one of several DB2 subsystems. The situation with CICS/ESA Version 3 is:

- CICS connects to DB2A.
- DB2A terminates abnormally, leaving an in-doubt.
- CICS connects to DB2B.
- The CICS attachment facility tells CICS that DB2 has no in-doubts to process.
- CICS does not distinguish between different DB2s; so it writes an RMI Forget record for the in-doubt on DB2A.
- When CICS later connects to DB2A, and DB2A sends the in-doubt to CICS, CICS replies that it should not be in-doubt.

Since all DB2 systems use the task related user exit (TRUE) name of DSNCSQL, CICS/ESA Version 3 sees all DB2 subsystems as the same resource manager.

The CICS/ESA Version 4 CICS attachment facility now adds the DB2 subsystem ID to the CICS recovery qualifier to distinguish one DB2 from another. Now during recovery, CICS deals only with the in-doubts for the specified DB2; so it remembers in-doubts across different DB2 connections.

9.4.4 Manual Resolution of In-doubt URs

In-doubt URs can occur when CICS, DB2, or the whole system fails while a transaction is carrying out syncpoint processing. In other words, a transaction started the syncpoint process using an EXEC CICS RETURN or EXEC CICS SYNCPOINT command, but that command had not completed processing at the time of the failure. In-doubt URs can also occur when the CICS attachment facility is terminated using a DSNC STOP FORCE command.

DB2 cannot resolve CICS in-doubt URs (that is, DB2 cannot commit or roll back the changes made to DB2 resources) until the connection to CICS is restarted. Once the connection between CICS and DB2 is reestablished, CICS and DB2 exchange information about the in-doubt URs so that they can be resolved. If CICS failed, CICS gets the necessary information from its logs during emergency restart. If you perform a cold start of CICS, the logs are not used, and the information needed for an automatic recovery is lost. For this reason you should never cold start CICS, but should always start it by specifying START=AUTO in the SIT, or as a startup parameter.

Under some circumstances, CICS cannot perform a DB2 in-doubt UR resolution process. When this happens, message DSNC001I, DSNC034I, DSNC035I or DSNC036I is sent to the user-designated CICS destination specified in the ERRDEST option of the TYPE=INIT macro in the RCT. The corresponding CICS/ESA 4.1 messages are DSN2001I, DSN2034I, DSN2035I, and DSN2036I. This section deals with operator actions related to this situation. Usually, URs are recovered automatically during connection or subsystem restart, so this is an exception procedure.

You must manually resolve any in-doubt UR that CICS cannot resolve by using DB2 commands. You should need to use this manual procedure only rarely,

since it is required only where operational errors or software problems prevent automatic resolution.

Any inconsistencies found during in-doubt resolution must be investigated. Failure to investigate these now could lead to a corrupt database in the future.

Use the following process:

1. Obtain a list of the in-doubt URs from DB2.

   To obtain this list, issue a DB2 DISPLAY THREAD command. Figure 88 shows the format of the command, as well as the messages issued.

```
DSNC -DISPLAY THREAD(connection-name) TYPE(INDOUBT)



DSNV401I  S  DISPLAY THREAD REPORT FOLLOWS:
DSNV406I  S  INDOUBT THREADS:

  NAME      ID           PLAN         NID
  name      corr-id      pname        net-node.number
  name      corr-id      pname        net-node.number

DISPLAY INDOUBT REPORT COMPLETE
DSN9022I  S  DSNVDT DISPLAY THREAD NORMAL COMPLETION
```

Figure 88. Output Format for DB2 Display In-doubt Thread Command

The parts of the messages shown in Figure 88 are explained below:

S is the DB2 subsystem recognition character.
name is a 1- to 8-character variable representing the ***connection name*** used to establish the thread. For CICS, it is the VTAM application name under which CICS was defined. This name is also used in the SIT APPLID option.
corr-id is a 1- to 12-character variable representing the recovery correlation ID associated with the thread. See Figure 89 on page 186 for details.
pname is a 1- to 8-character variable representing the plan name associated with the thread.
net-node.number is a 1- to 25-character variable representing the recovery network ID associated with the in-doubt thread. It is the external representation for displaying in-doubt threads.

When the connection is broken several times and the in-doubt URs are not resolved, two threads can have the same correlation ID. In this case, the network ID (NID) must be used instead of the correlation ID to uniquely identify in-doubt URs. The network ID consists of the CICS connection name and a unique number provided by CICS at the time the syncpoint log entries are written. This unique number is an 8-byte store clock value that is stored in records written to both the CICS system log and to the DB2 log at syncpoint processing time. This value is referred to in CICS as the *recovery token*.

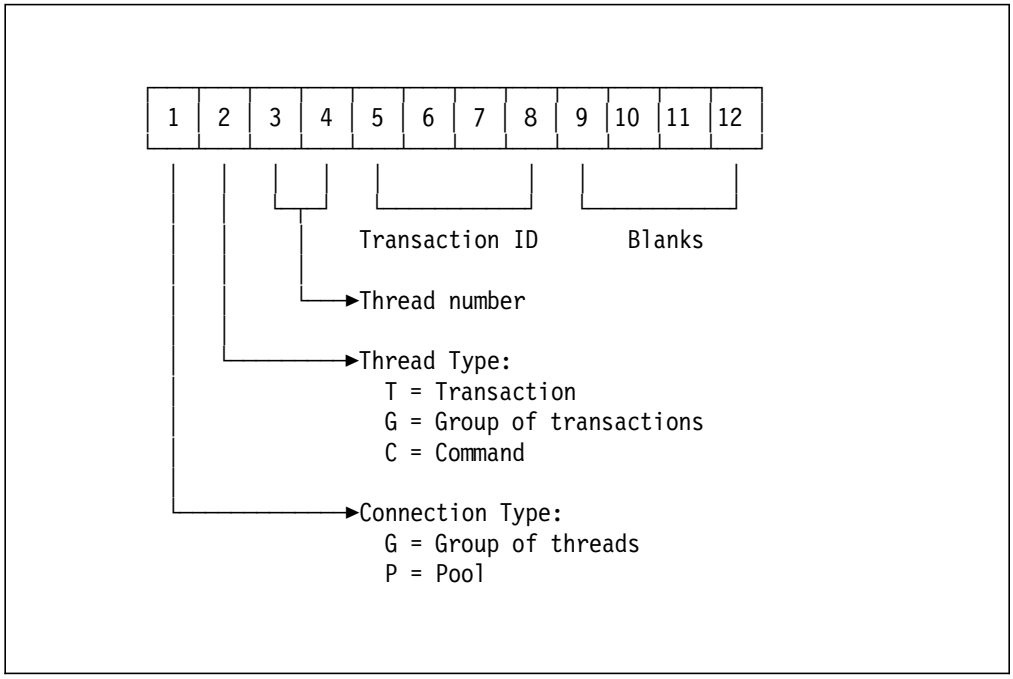


Figure 89. Format of the Correlation ID

2. Obtain a list of the in-doubt URs from CICS.

   Use the CICS utility DFH$INDB to obtain a list of the in-doubt external resource records from CICS, and the recovery action you should take. The DFH$INDB utility is described in the *Operations Guide* for your version of CICS.

3. If needed, perform in-doubt resolution in DB2.

   You can direct DB2 to take the recovery action for an in-doubt UR using a DB2 RECOVER INDOUBT command. If the correlation ID is unique, use the form of the command as shown in Figure 90.

```
DSNC -RECOVER INDOUBT (connection-name)  -
             ACTION (COMMIT/ABORT)       -
             ID(correlation ID)
```

Figure 90. Using the RECOVER INDOUBT Command: Unique Correlation ID

Figure 89 shows that the correlation ID is made up of five components:

- The connection type
- The thread type
- The thread number
- A transaction ID
- Four blanks.

If the transaction is associated with a pool thread, the correlation ID (*corr-id* in Figure 88 on page 185) returned by -DISPLAY THREAD that you use in the -RECOVER INDOUBT command has a P as its first letter. The transaction ID is in bytes 5 through 8 of the correlation ID.

If the transaction is assigned to a group (group is a result of using an entry thread), the correlation ID has a G as its first letter. The transaction ID in

bytes 5 through 8 of the correlation ID is the *groupname* (the first transaction listed in the TYPE=ENTRY statement in the RCT), rather than the actual transaction ID of the transaction using this thread.

If the correlation ID is not unique, use the form of the command shown in Figure 91.

```
DSNC -RECOVER INDOUBT (connection-name)  -
             ACTION (COMMIT/ABORT)       -
             NID(network-id)
```

Figure 91. Using the RECOVER INDOUBT Command: Nonunique Correlation ID

If DB2 reports an in-doubt UR and CICS does not, the correct action is to back out the UR. In this case DB2 wrote its prepare record to the log before CICS did. The commit process was still in Phase 1, and the correct action is to back out the UR.

If CICS reports an in-doubt UR and DB2 does not, you can ignore the discrepancy. In this case the UR is completed, but the failure occurred before CICS could write the RMI Forget record to its log.

Note: CICS and the CICS attachment facility are not aware of the commands to DB2 to commit or back out in-doubt URs, since only DB2 resources are affected. However, CICS keeps details about the in-doubt threads that could not be resolved by DB2. This information is purged:

- When the list presented by DB2 is empty
- When the list does not include a UR that CICS remembers.

See the section discussing manually recovering CICS in-doubt units of recovery in the *IBM DATABASE 2 Administration Guide* for further details.

9.5 CICS Recovery and Restart Specifications

Before reading this section, note that there are no DB2 recovery and restart specifications. DB2 forces recovery of any data under its control.

CICS recovery and restart specifications are related to:

- The recovery of CICS controlled resources (that is, VSAM files, transient data queues, temporary storage queues). Follow the *CICS/ESA Recovery and Restart Guide* to specify recovery for any of these resources. These specifications are not affected by the CICS attachment facility.

  Note that both CICS and DB2 have a built-in dynamic transaction backout function; so if a transaction updates CICS and DB2 resources the CICS and DB2 updates are synchronized if the transaction abends.

- DB2 resources are not defined to CICS; so there are no CICS recovery and restart specifications related to them.

- Even if no CICS resources are being updated, still some CICS recovery and restart specifications are needed when connecting to DB2. When there is a CICS failure, CICS restart builds the in-doubt resolution list from the log. At a minimum, the following specifications are required for providing this support:

- A journal control program
- A keypoint program
- A journal control table (JCT) with an entry for the system log.
- The CICS-supplied groups DFHAKP, DFHBACK, DFHJRNL, and DFHSTAND are included in the LIST you use when you cold start CICS.

If these specifications are missing, CICS cannot perform in-doubt resolution in the case of a failure, forcing you to resolve manually if DB2 has in-doubt URs. Of course, the resolution process in this case is simplified by the fact that no CICS recoverable resources were updated.

9.6 Application Design Considerations for Recovery and Restart

The *CICS/ESA Recovery and Restart Guide* covers the application design considerations related to recovery and restart. There are no special complications when a transaction updates data on both subsystems, since CICS and DB2 ensure synchronization of commit processing between the two subsystems. Design your application so that required updates are performed in step with each other and are:

- In the same CICS LUW, or
- In the same DB2 UR, or
- In a CICS LUW and in a DB2 UR that are synchronized.

Figure 92 shows an example of update synchronization.

See also Chapter 10, “CICS-DB2 Application Design Considerations” on page 191 for additional recommendations on how to design DB2-oriented CICS application programs.

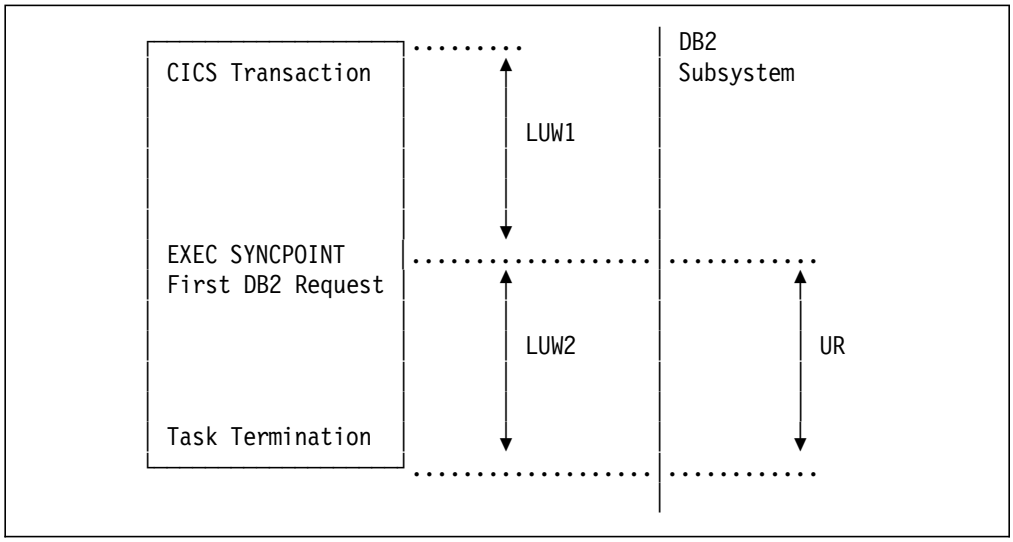


Figure 92. Synchronization of Commit Protocols

Notes on Figure 92

Here, the CICS transaction uses EXEC CICS SYNCPOINT to commit before accessing DB2; so the transaction is divided into two LUWs, while there is only one DB2 unit of recovery (UR).

LUW1 updates only CICS resources and it is not synchronized with the DB2 UR. A backout after the SYNCPOINT backs out LUW2 and the DB2 UR, but not LUW1.

9.6.1 Transaction Restart

Specifying specify RESTART=YES in your transaction definition requests the use of the transaction restart facility to restart those tasks that terminate abnormally. They are subsequently backed out by the dynamic transaction backout facility. In the discussion of conditions required for restarting a task after DTB, the *CICS/ESA Recovery and Restart Guide* states what CICS does. If specific conditions are not all met, DFHDBP sets the indicator to tell CICS not to attempt restart, but still passes control to the transaction restart program (DFHRTY).

With a CICS-DB2 transaction, this indicator is always set to say that restart should *not* be attempted. Nevertheless, it is your responsibility to design your own DFHRTY, at installation level:

- To determine if the CICS-DB2 transaction can be restarted
- To check for possible recursive restarts.

9.6.2 Point-in-Time Recovery Considerations

DB2 provides a point-in-time recovery facility. To use it in a CICS environment, you should design your application to take advantage of it. Thus, if you choose to recover the DB2 data to an earlier point in time:

- All DB2-related tablespaces should be copied and kept in sync.
- The DB2 tablespaces should be recovered to the same level.
- Other CICS resources (VSAM clusters, and so on) should be copied and recovered to the same point as DB2 data, to keep all related data consistent.

Chapter 10. CICS-DB2 Application Design Considerations

This chapter contains information primarily for the CICS application developer, the CICS application reviewer, and those involved in defining standards for application design. This chapter is divided into the following sections:

- Overview
  Explains the importance of a consistent application design.
- CICS-DB2 Design Criteria
  Discusses how the CICS user should use the functions and options that are unique for DB2 and the CICS-DB2 interface.
- Application Architecture
  Describes different ways of implementing the application in CICS transaction codes and DB2 plans. Also discusses some consequences of the methods.
- Programming.
  Describes a number of programming-related aspects that apply to the CICS-DB2 environment.

Note that this chapter deals only with the design recommendations that are unique to the CICS and DB2 environments. The general considerations that apply only to DB2 applications are not covered in this chapter.

See the *IBM DATABASE 2 Application Programming and SQL Guide* for information on DB2 application design and the *CICS/ESA Application Programming Guide* for information on CICS application design.

10.1 Overview

Using DB2 as the DBMS is advantageous in most steps of the CICS development process and in the production environment.

However, the users involved in designing and developing CICS-DB2 applications must be aware of several differences between those applications and applications developed with data stored in VSAM and DL/I. Some of the main differences to consider are:

- Locking mechanism
- Security
- Recovery and restart
- BIND process
- Operational procedures
- Performance
- Programming techniques.

To address these differences in the CICS application developing process, the first-time user needs some guidelines.

One of the major differences between batch and online program design is that online systems should be designed for a high degree of concurrency. At the same time, no compromise should be made to data integrity. In addition, most online systems have design criteria about performance.

10.2 CICS-DB2 Design Criteria

In the design process, decisions can be taken that have consequences related not only to the application being developed now, but also to future applications. Some of the key decisions deal with the:

- Use of qualified and unqualified SQL
- Locking strategy
- Dependency of specific BIND options
- Dependency of specific RCT parameters
- Security principles
- Transaction code usage
- When to switch plans
- Programming standards.

In addition, the user should consider that the design being implemented can influence:

- Performance
- Concurrency
- Operation
- Security
- Accounting
- Development environment.

The applications are likely to work properly when set into production for the first time. However, certain occurrences can lead to situations where some of the consequences described above are negative. These occurrences include:

- Increased transaction rate
- Continued development of existing applications
- More people involved in developing CICS-DB2 applications
- Existing tables used in new applications
- Integration of applications.

It is therefore important to develop a consistent set of standards on using DB2 in the CICS environment.

The following sections discuss the key decisions regarding CICS-DB2 application development.

10.2.1 Using Qualified and Unqualified SQL

Programmers writing CICS-DB2 programs can use qualified and unqualified SQL. In qualified SQL, the creator is specified in front of the table or view name. In unqualified SQL, the creator is not specified.

When programmers develop CICS-DB2 standards, it is important to determine the use of qualified and unqualified SQL. This decision influences many other aspects of the DB2 environment. The main relationships to other DB2 areas are shown in Figure 93 on page 193.

	Qualified SQL	Unqualified SQL
Use of synonyms	Not possible	Possible
Binder ID	Any	Same as creator
Number of creators for tables and table spaces	Any	One
Use of VALIDATE(RUN)	Is qualified	Uses binder to qualify
Use of dynamic SQL	Is qualified	Uses executor to qualify
Require a separate test DB2 subsystem	Yes	No
Require same creator in test DB2 and production DB2	Yes	No
Possibility of using multiple versions of the test tables in the same test DB2 subsystem	No	Yes

Figure 93. Qualified and Unqualified SQL. The table shows some consequences for the two types of SQL statements.

Some of the limitations shown in Figure 93 can be bypassed, if you develop your own preprocessor to modify the source code before invoking the DB2 precompiler. This allows you, for example, to change the creator in the SQL statements.

10.2.2 Locking Strategy

The main purpose of the lock mechanism in DB2 is to allow concurrency while maintaining data integrity.

In a CICS environment, concurrency is likely to be high. To give maximum concurrency, you should use page locking instead of table space locking. You can do this by defining LOCKSIZE(PAGE) or LOCKSIZE(ANY) when creating the table space and by defining the isolation level as cursor stability at BIND time.

Specifying LOCKSIZE(ANY) allows DB2 to decide if lock escalation can take place for the table space. If the number of locks exceeds NUMLKTS, lock escalation takes place. The DB2 parameter NUMLKTS is the number of concurrent locks for a table space. NUMLKTS should then be set to a value so high that lock escalation cannot take place for normal CICS operation.

Using ANY instead of PAGE gives DB2 the option to use lock escalation for programs that require many page locks before committing. This is typically the case in batch programs.

In general, we recommend designing CICS programs so that:

- Locks are kept as short as possible.

- The number of concurrent locks is minimized.
- The access order of the tables is the same for all transactions.
- The access order of the rows inside a table is the same for all transactions.

The LOCK TABLE statement should not be used, unless specifically needed. If you do use the LOCK TABLE statement, your plan should use the bind option RELEASE(COMMIT).

10.2.3 Bind Options

When you bind a plan, a number of options are available. You should develop procedures to handle different BIND options for different plans. Also the procedures should be able to handle changes in BIND options for the same plan over time.

We make specific recommendations for BIND options in CICS below.

10.2.3.1 Isolation Level

We recommend you use *Cursor Stability (CS)* unless there is a specific need for using Repeatable Read (RR). This is recommended to allow a high level of concurrency and to reduce the risk of deadlocks.

Note that the isolation level is specified for the complete plan. This means that if RR is necessary for a specific module in CICS, then all the DBRMs included in the plan must also use RR.

Also, if for performance reasons you decide to group a number of infrequently used transactions in the RCT and let them use a common plan, then this new plan must also use RR, if just one of the transactions requires RR.

10.2.3.2 Plan Validation Time

A plan is bound with VALIDATE(RUN) or VALIDATE(BIND). VALIDATE(RUN) is used to determine how to process SQL statements that cannot be bound.

If a statement must be bound at execution time, it is rebound for each execution. This means that the statement is rebound for every new unit of recovery.

Binding a statement at execution time can affect performance. A statement bound at execution time is rebound for each execution. That is, the statement must be rebound after each commit point. We do not recommend using this option in CICS.

Note that using dynamic SQL does not require VALIDATE(RUN). Nevertheless, dynamic SQL implies that a statement is bound at execution.

You should use VALIDATE(BIND) in a CICS-DB2 environment.

10.2.3.3 ACQUIRE and RELEASE

The general recommendations for these parameters are in section 5.3, “Coordinating RCT and BIND Options” on page 80. The parameters change from plan to plan and over time, because they are related to the transaction rate.

10.2.4 Using Protected Threads

Using protected threads is a performance option that reduces the resources involved in creating and terminating a thread. For performance reasons, we recommend using protected threads whenever possible. See Chapter 5, “Defining the Resource Control Table” on page 75 for more information about the performance advantages of using protected threads.

From an accounting viewpoint, the situation is different. An accounting record is produced for each thread termination and for each new user sign-on. This means that only one accounting record is produced if the thread stays alive and the user ID does not change. This record contains summarized values for all transactions using the same thread, but it is not possible to assign any value to a specific transaction. See Chapter 8, “Accounting” on page 153 for more information about accounting in a CICS-DB2 environment.

10.2.5 Security

When users define the DB2 standards, the security aspects of both the CICS-DB2 test system and the CICS-DB2 production system can be important.

When reusing a thread, DB2 checks the user′s authorization and whether the user ID changed from the previous transaction using the thread. From a performance viewpoint, the user ID should not change from one transaction to another.

The major considerations concerning security in a CICS-DB2 system are described in Chapter 4, “Security” on page 61.

10.2.6 Dynamic Plan Switching

In DB2, you can design CICS applications around numerous small plans and select the plan dynamically at execution time. (A small plan is not the same as a package, which has a strictly one-to-one correspondence to a database request module (DBRM)). DB2 plan allocation occurs only upon execution of the first SQL statement in a program, or after the program issues a syncpoint and links or transfers control to another program with a separate DBRM.

This is accomplished by using an exit program designated in the RCT specified in:

- TYPE=INIT, default exit for all threads, in the keyword ***PLNPGMI***
- TYPE=ENTRY, exit for a specific transaction code specified in the keyword ***PLNPGME***
- TYPE=POOL, exit for transactions not using a TYPE=ENTRY definition. The EXIT program is specified in the keyword ***PLNPGME***.

  IBM supplies a sample exit program in assembler language and object code. You can also write other exit programs.

10.2.6.1 Exit Program

The exit program can be written in assembler language, COBOL, or PL/I. The program is a CICS program, which means it must:

- Adhere to normal CICS conventions
- Be defined to CICS
- Use CICS command level statements

- Return to CICS using an EXEC CICS RETURN command.

The CICS attachment facility program passes a parameter list to the exit program using a COMMAREA. The exit program can change the default plan name (DBRM name) supplied in the parameter list, when the first SQL statement is processed by the CICS attachment facility. The name specifies the plan name for this execution of the transaction.

10.2.6.2 Sample Exit Program

An object code sample, DSNCUEXT, is included in DB2 2.3, DB2 3.1, and CICS/ESA Version 4. The source for this sample user exit program is in library DSN230.DSNSAMP, DSN310.SDSNSAMP, or CICS410.SDFHSAMP, with the member name DSNC@EXT. The supplied source code is written in assembler language. The sample program shows how to address the parameter list and does not change the plan name. Refer to the discussion of writing exit routines in *IBM DATABASE 2 Administration Guide* for details on this exit.

10.2.6.3 Implementation

Dynamic plan switching (DPS) allows you to use more than one plan in a transaction. However, switching plans within a CICS transaction instance should be a rare occurrence. Dynamic plan switching was designed to select a plan dynamically, not to change plans frequently within transactions.

To do dynamic plan switching:

- The thread must be a pool thread.
- The thread must be released at syncpoint and reacquired to drive the dynamic plan selection exit.

A transaction running on a protected entry thread calls the dynamic plan exit once, on the first SQL call of the transaction. Syncpoints have no effect on this. Non-terminal transactions running on unprotected entry or pool threads have the same effect because they call the dynamic plan exit only on the first SQL statement in the transaction.

To invoke the dynamic plan exit to do plan switching after the end of a unit of recovery, your transaction must release the thread at syncpoint. A transaction releases a thread at syncpoint only if:

- It is a terminal driven task, *AND*
- *NO* held cursors are open, *AND*
- *NO* modifiable special registers are in use.

In addition, the thread must be a pool thread to redrive the dynamic plan exit at syncpoint.

To divert an unprotected entry thread to the pool, code the RCT entry with

```
TYPE=ENTRY,THRDM=0,THRDA=0 and TWAIT=POOL
```

To assign the transaction directly to a pool thread, do not code the RCT entry at all. The transaction then defaults to a pool thread. Its plan name must be selected by the exit routine rather than having the fixed pool value of PLAN=planname.

10.2.7 Packages

In DB2 2.3 and later, you can probably more easily obtain the function provided by dynamic plan switching by using packages.

Dynamic plan switching was intended to ease two problems that occur, for a program running under a CICS transaction, when all SQL calls are bound together into a single plan. First, changing one DBRM in a plan requires all the DBRMs in the plan to be bound again. Second, binding a large plan can be very slow, and the entire transaction is unavailable for processing during the operation. Just quiescing an application can be very difficult for a highly used one, but to leave the plan out of commission for an extended time while it is being rebound is usually unacceptable.

With packages, you can break the program units into much smaller parts, which you can rebind *individually* without affecting the entire plan or even the current users of the particular package you are rebinding.

Since updating the plan is easier with packages, you can build much larger applications without the need to switch transactions, programs, or plans to accommodate DB2 performance or availability. This also means that you do not have to maintain as many RDO or attachment RCT entries. You can also avoid situations where you might otherwise use dynamic SQL for program flexibility. Programs are easily expanded by specifying packages that do not exist yet or specifying package collections.

The qualifier option on packages and plans to reference different table sets can give you more flexibility to avoid plan switching.

In summary, packages

- Minimize
  - Plan outage time
  - Processor time
  - Catalog table locks

  during bind for programs that are logically linked together with START, LINK, or RETURN TRANSID and have DBRMs bound together to reduce RCT entries.
- Reduce CICS STARTS or exits
- Avoid cloning CICS and RCT table entries
- Provide the ability to bind a plan with null packages for later inclusion in the plan
- Allow you to specify collection at execution time
  - SET CURRENT PACKAGESET = `variable`, which is a powerful feature when used with QUALIFIER
- Provide the QUALIFIER parameter, which adds flexibility to
  - Allow you to reference different table sets using unqualified SQL
  - Reduce the need for synonyms and aliases
  - Lessen the need for dynamic SQL.

Other benefits that using packages can provide in reference to RCT:

- Reduces RCT reassembly frequency
- Decreases the number of RCT entries
- Allows you to use fewer plans
- Allows you to bind low-use transactions into a single plan
- Increases thread reuse potential.

You can also optimize your application by using package lists that order their entries to reduce search time or by keeping package list entries to a minimum.

DB2 3.1 also provides accounting at the package level. For more information about packages, refer to the discussions of planning to bind and preparing an application program to run in *IBM DATABASE 2 Application Programming and SQL Guide* and *DB2 Packages: Implementation and Use*.

10.2.8 Avoiding AEY9 Abends

With CICS/ESA Version 3, you can use the following CICS command to detect whether the CICS attachment facility is enabled:

```
EXEC CICS EXTRACT EXIT PROGRAM('DSNCEXT1')
          ENTRY('DSNCSQL')
          GASET(name1)
          GALENGTH(name2)
```

If you get the INVEXITREQ condition, then the CICS attachment facility is not enabled. However, there is a short window between the CICS attachment facility's enable and the enable start where the CICS attachment facility is still not available. During that period, you can get an AEY9 abend.

INQUIRE EXITPROGRAM is a new command that CICS/ESA Version 4 provides. You can use it in place of the EXTRACT EXIT command to determine whether the CICS attachment facility is enable-started, enabled, or disabled. If the attach is enabled without being enable-started, you can still get an AEY9 abend, and the EXTRACT EXIT command can tell you only if the CICS attachment facility is enabled.

The CONNECTST keyword of the INQUIRE EXITPROGRAM command has three values:

- NOTAPPLIC, when there is no DB2 or DBCTL
- CONNECTED, when the CICS attachment facility is ready to accept SQL requests
- NOTCONNECTED, when the CICS attachment facility is not ready to accept SQL requests.

Figure 94 on page 199 shows an example of assembler code using the INQUIRE EXITPROGRAM command.

```
CSTAT    DS    F
ENTNAME  DS    CL8
EXITPROG DS    CL8
         ...
         MVC   ENTNAME,=CL8'DSNCSQL'
         MVC   EXITPROG,=CL8'DSN2EXT1'
         EXEC CICS INQUIRE EXITPROGRAM(EXITPROG)               X
               ENTRYNAME(ENTNAME) CONNECTST(CSTAT) NOHANDLE
         CLC   EIBRESP,DFHRESP(NORMAL)
         BNE   NOTREADY
         CLC   CSTAT,DFHVALUE(CONNECTED)
         BNE   NOTREADY
```

Figure 94. Example of the INQUIRE EXITPROGRAM Command

10.2.9 CICS and CURSOR WITH HOLD Option

The WITH HOLD option on a CURSOR declaration in a CICS program causes the following effects during a COMMIT or SYNCPOINT:

- The cursor is kept open.
- The cursor is left in position after the last row which was retrieved, and before the next row in the results table.
- Dynamic SQL statements are still prepared.

All locks are released, except for those required to maintain the cursor's position. Any exclusive page locks are downgraded to shared locks.

In conversational CICS applications, you can use DECLARE CURSOR...WITH HOLD to request that the cursor is not closed at commit or syncpoint time. However, all cursors are *always* closed at end of task (EOT) and on SYNCPOINT ROLLBACK. Across EOTs, a cursor declared WITH HOLD must be reopened and repositioned just as if the WITH HOLD option were not specified. The scope of the held cursor is a single task.

In summary :

- The next FETCH following a syncpoint must come from the same task.
- You cannot hold a cursor across end of task.
- Therefore, cursors are *not* held across the EOT portions of pseudoconversational transactions.

If you try to hold a cursor across EOT, the cursor is closed and you get an SQLCODE -501 when you execute the next FETCH. Of course, the precompiler cannot prevent this and you do not get a warning message notifying you of this situation.

In general, threads can become candidates for reuse at each syncpoint. When you use DECLARE CURSOR...WITH HOLD in the CICS applications, consider the following recommendations:

- Close held cursors as soon as they are no longer needed. Once all held cursors are closed, syncpoint can free the thread for thread reuse.

- Always close held cursors before EOT. Terminal-driven transactions can only free threads at syncpoint if all held cursors are closed and no modifiable special registers are in use.

If you do not close your held cursors, the CICS attachment facility forces sign-on to restore the thread to the initial state, and this incurs additional processor time.

10.2.10 RETURN IMMEDIATE Command

CICS/ESA 3.2.1 introduced some additional options on the EXEC CICS RETURN command. The new options are IMMEDIATE, INPUTMSG, and INPUTMSGLEN. When the existing TRANSID option is specified in conjunction with the new IMMEDIATE option, CICS avoids sending an end bracket (EB) to the terminal during the termination of the transaction that issued the RETURN command, and immediately initiates the transaction designated by the TRANSID option. The keyboard remains locked during this transaction, since no EB was sent to the terminal.

The new transaction behaves as if it were started by input from the terminal. You can pass data to the transaction designated by the TRANSID option, using a COMMAREA. If you choose to, the transaction issuing the RETURN command can also pass a terminal input message using the INPUTMSG and INPUTMSGLEN options. This facility allows you to immediately initiate a transaction that expects to be initiated as a result of terminal input.

This facility provides the same general capability as that achieved by issuing an EXEC CICS START TRANSID(....) with TERMID(....) set to the EIBTRMID value, but with much less overhead and without the momentary keyboard unlocking. The EXEC CICS RETURN TRANSID() IMMEDIATE command permits a pseudoconversational transaction to switch transaction codes. This could be desirable, for example, to keep DB2 plan sizes smaller or to have better accounting statistics for charge-back purposes.

10.3 Application Architecture

When CICS applications use DB2 data, the application architecture must take account of design aspects related to the CICS attachment facility. One of the most important aspects to consider is the relationship between transaction IDs, DB2 plans and packages, and program modules. If any of the program modules uses SQL calls, a corresponding DB2 plan or package must be available. The plan to be used for a transaction ID is defined in the RCT. The plan can be named explicitly, or we can name a plan exit routine that selects the plan name for us. The plan must include the DBRMs from all modules that could possibly run under this transaction ID.

To control the characteristics of the plan and the CICS attachment facility threads, the relationship between transaction IDs, DB2 plans, and the program modules must be defined in the design step. Some characteristics of the threads, environmental description manager (EDM) pool, and plans that depend on the design are the:

- Plan sizes
- Number of different plans concurrently in use
- Number of threads concurrently in use
- Possibility of reusing a thread

- Size of the EDM pool
- I/O activity in the EDM pool
- Overall page rate in the system
- Authorization granularity
- Use of DB2 packages.

These characteristics in turn influence the factors listed in 10.2, “CICS-DB2 Design Criteria” on page 192.

10.3.1 A Sample Application

We use a simple example to explain the consequences of different application design techniques. Figure 95 shows how CICS MAPs and transaction IDs are correlated.

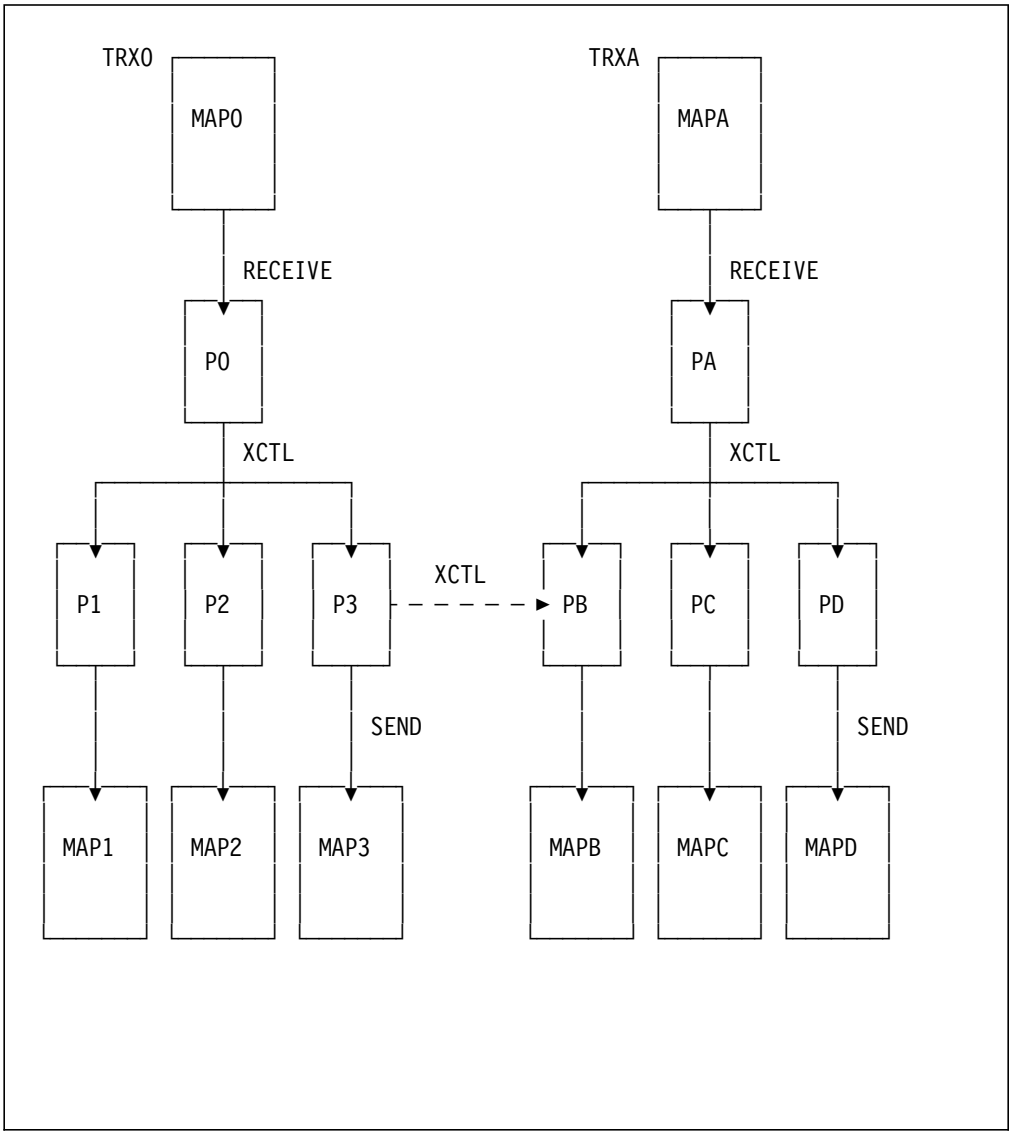


Figure 95. Example of a Typical Application Design. The figure shows how the transactions should work, without DB2 considerations.

In this example:

- The transaction ID TRX0 is specified in the EXEC CICS RETURN TRANSID(TRX0) command, when a program (not shown) returns control after displaying MAP0.
- The next transaction then uses the transaction ID TRX0, independent of what the terminal user decided to do.
- Program P0 is the initial program for transaction TRX0.
- We assume that all programs shown are issuing SQL calls.
- Depending on the option chosen by the terminal user, program P0 performs a CICS Transfer Control (XCTL) to one of the programs: P1, P2, or P3.
- After issuing some SQL calls, these programs display one of the maps: MAP1, MAP2, or MAP3.
- The example shown on the right side of the figure works in the same way.
- In some situations, program P3 transfers control to program PB.

Later sections discuss the following design techniques for combining CICS transactions with DB2 plans:

- One large plan
- Many small plans
- Transaction grouping
- A fallback solution
- Table-controlled program flow
- Using packages.

First, however, we discuss the general problem with plan switching.

10.3.2 Switching CICS Transaction Codes

If you do not use packages or dynamic plan switching, the program design can often dictate a plan switch when a program not included in the current plan must be executed. The programmer does not control the plan, only the transaction ID. So, from a programmer viewpoint, the transaction IDs must be switched. We strongly recommend that you use packages (and bind the DBRM for this program into the existing plan) rather than changing the transaction ID just to switch to a new plan.

In most cases, the first program transfers data to the next program. The preferred method of doing this is to use an EXEC CICS RETURN IMMEDIATE command. Alternatively you can start a new CICS task against the same terminal using an EXEC CICS START command or using a transient data queue with a trigger level of one. The old program should issue RETURN to CICS to make the new task start. For both of these switching techniques, the work done in one user transaction is split up into more than one LUW. If the new task is backed out, the work done in the first task remains committed.

10.3.3 One Large Plan

Using one large plan for all CICS transactions that issue SQL calls is an easy strategy. For the example in Figure 95 on page 201, this technique implies:

- One plan, PLAN0, using the DBRMs from P0, P1, P2, P3, PA, PB, PC, and PD
- The RCT could have one or two entries, both using PLAN0. One entry gives the best overall thread utilization if protected threads are used.

10.3.3.1 Advantages

- No restrictions regarding which program modules can be executed under any transaction ID. For example, it is possible that program P3 can transfer control to program PB. This does not require any changes for the plan or the RCT.
- Each DBRM exists in only one plan (PLAN0).
- Simple to maintain. Any DBRM modification requires only that this plan be BOUND again.
- Thread reuse is easy to obtain. All transactions could use the same RCT entry.

10.3.3.2 Disadvantages

- The complete plan must be rebound for any DB2 program modification.
- BIND can be time-consuming for large plans.
- The BIND process cannot take place while the plan is in use. The plan is likely to be in use in a production system most of the time due to normal activity. In a test environment, the transaction rate is normally low, but programmers can use debugging tools that make the response times longer with conversational programs. This can effectively keep the thread and plan busy.
- DB2-invoked REBIND (due to plan invalidation) allows no DB2 transactions to execute this plan.
- There is no real information value in messages and statistics pointing out the plan name, because there is only one plan.
- EDMPOOL must be large enough to cope with DBDs, SKCTs, and CTs and must allow some fragmentation. Remember that starting with DB2 Release 2, the plan segmentation feature allows DB2 to load into CTs only parts of the application plans being executed. Nevertheless, the header and directory parts of an application plan are loaded in their entirety into the SKCT (if not already loaded), then copied from the SKCT to CTs. This happens at thread creation time.

  Because the application plan directory size is directly dependent on the number of segments in the plan, using a large plan influences the EDMPOOL size and the number of I/O operations needed to control it. See the discussions of DB2 transaction flow and EDM pool in *IBM DATABASE 2 System Monitoring and Tuning Guide*.

10.3.4 Many Small Plans

This technique requires many transaction IDs, each of which has a corresponding plan. The technique can minimize both plan sizes and plan overlap.

We recommend that you use packages rather than many small plans.

Using many small plans implies either that the program flow follows a narrow path with limited possibilities for branching out, or that plan switching takes place frequently.

In the example in Figure 95 on page 201, the switching could take place between program P0 and the programs at the next lower level, or between program PA and the programs at the next lower level.

- PLAN1 for (TRX0) using the DBRMs from programs P0, P1, P2, and P3.
- PLANA for (TRXA) using the DBRMs from programs PA, PB, PC, and PD.

However, program P3 can transfer control (using the XCTL command) to program PB. A plan switching technique must then be used. These techniques are described in section 10.3.2, “Switching CICS Transaction Codes” on page 202.

A special variation of using small plans exists. In some applications, it can be convenient to have the terminal user specify the transaction ID for the next transaction. It is typically in read-only applications, where the user can choose between many different information panels by entering a systematically built 1- to 4-character transaction ID. The advantage for the user is the ability to jump from any panel to any other panel without passing a hierarchy of submenus.

If a DB2 plan is associated with each transaction ID, the application ends up with many small plans.

10.3.4.1 Advantages

- Plan maintenance is relatively simple, because little overlap exists between plans.
- High information value in messages, statistics, and so on, pointing out the plan-name.

Note: Packages offer these advantages, too.

10.3.4.2 Disadvantages

- Plan switching occurs often, unless the application flow follows a narrow path.
- It is difficult to use protected threads, because the transactions are spread over many sets of transaction IDs, plans, and threads.
- Resource consumption can be high, due to plan switching and low thread reuse.

Note: Packages avoid these disadvantages.

10.3.5 Transaction Grouping

Transaction grouping can produce a number of midsize independent plans, where a plan switch can occur if necessary.

It is often possible to define such groups of programs, where the programs inside a group are closely related. That means that they are often executed in the same transaction, or in different transactions being executed consecutively. One separate plan should then be used for each group.

In the example in Figure 95 on page 201 two plans could be built:

- PLAN1 for (TRX0) using the DBRMs from programs P0, P1, P2, and P3.
- PLANA for (TRXA) using the DBRMs from programs PA, PB, PC, and PD.

However, program P3 can transfer control to program PB. A plan switching technique could then be used. These techniques are described in section 10.3.2, “Switching CICS Transaction Codes” on page 202. We recommend that plan switching is an exception and not the normal case, mainly due to additional processor overhead.

In this case, the result of the transaction grouping technique matches the result for the technique of using many small plans. This is because of the simplicity of the example used. Normally the transaction grouping technique should produce a larger plan.

10.3.5.1 Advantages

- The plan size and the number of different plans can be controlled by the user.
- Thread reuse is likely, depending on the transaction rate within the group.

10.3.5.2 Disadvantages

- Plan overlap can occur.
- The optimum grouping can change over time.
- Plan switching can be necessary.
- Plan switching is handled in the application code.

10.3.6 A Fallback Solution

You can use this technique if the applications were developed with only little attention to the DB2 plan aspects. After the application is completely developed, the plans are defined to match the transaction.

We do not recommend this technique in general, but it is useful for *conversion projects*, where the application design is unchanged but the application now uses DB2.

When defining the DB2 plans and the RCT specifications, you can perform the following steps:

1. For each program module with SQL calls, analyze under which CICS transaction codes they might run. It is likely that a given program module will be used under more than one CICS transaction code.

   The output from this step is a list of DBRMs for each transaction ID.

2. For each CICS transaction code decide which plan it should use. (Only one plan may be specified in the RCT for a given CICS transaction code. More than one transaction may use the same plan).

   For this step you have many alternatives. The possible number of plans to use is between one and the number of different transaction IDs.

   A procedure could be to define a plan for each transaction ID and estimate the plan sizes. You can use the techniques described in section 5.7, “Sample RCT Specification” on page 99 to decide if some of the plans should be grouped together.

Applied to the example in Figure 95 on page 201, the fallback solution implies:

- One plan, PLAN0, using the DBRMs from P0, P1, P2, P3, and PB, used by the transaction ID TRX0

- One plan, PLANA, using the DBRMs from PA, PB, PC, and PD, used by the transaction ID TRXA
- The RCT specified accordingly, with two entries.

The advantages and disadvantages of the fallback technique completely depend on the actual application design and the result of the above-mentioned steps.

10.3.7 Table-Controlled Program Flow

One of the main problems with the techniques discussed so far is that the programs must contain logic to decide when to switch plans (that is transaction ID). If you want to change the plan structure (for example for performance and operational reasons), you then need to change the application programs accordingly.

A different technique is to have the program flow controlled by a table instead of having code to do this in several programs. The main idea is that it is simpler to maintain the relationship between the plans, programs, and transaction IDs in a table than to maintain code in each application program to do the logic.

Developing the application programs is also simpler if a standard interface is provided.

10.3.7.1 Control Table

The control table is used to control the program flow in the transactions.

The design principle presented below is an example of a standard method that can be used to implement different types of application design. It allows the use of for example one large plan or many small plans *without* changing the programs.

Note: We recommend that you use packages rather than this technique to control program flow.

Figure 96 on page 208 shows an example of the contents in the control table. The example is based on the design situations described in Figure 95 on page 201.

Function name The Function name field of the table supplements the Program field. It works in exactly the same way. It is used in the cases where the terminal user enters a Function name in a command field, eventually supplied with a key. The the PFCP program can accept either a Function name or a Program name.

PFCP then uses the function column to search for a valid transaction ID.

In this way, the logic to interpret the user's choices is also removed from the programs and placed in the table, where it is easy to maintain.

Program The name of the program described in this row.

Transaction The transaction ID name under which the program in the Program name column can be executed.

Plan name The Plan name field is not used. It is shown for illustration purposes only. It shows the plan name used by the corresponding transaction.

New TRANS ID An * in this column of the table means that the corresponding row can be used when searching for a new transaction ID to start a given program.

The control table reflects the following main relationships:

- Relationships between plans and programs/DBRMs, which are described in the DB2 catalog table SYSIBM.SYSDBRM
- Relationships between transaction codes and plans, which are described in the RCT.

When implementing the control table, you should consider that:

- Many different RCTs can be used by a given CICS system over time, which means that different RCT names can be used at different times.
- The same RCT name can be used in different CICS systems, for example the RCT with suffix 0.

One method to handle these possibilities is to add two additional columns to the control table:

- RCT suffix
- CICS APPLID

At execution time the PFCP can search the control table for rows belonging to the current APPLID and RCT suffix. The RCT suffix for the RCT currently in use can be found by the following method.

To retrieve the address of DSNCVECT in name1, use the CICS command:

```
EXEC CICS EXTRACT EXIT
          PROGRAM(DSNCEXT1)
          ENTRY(DSNCSQL)
          GASET(name1)
          GALENGTH(name2)
```

Bytes 9-12 of DSNCVECT point to the RCT in use. Byte 8 of this RCT contains the suffix character.

The control table can be implemented in different ways. The most useful solutions are probably either a DB2 table or a main storage table.

A *DB2 table* is the simplest to develop and maintain. One or two SELECT calls are needed for each invocation of PFCP. These SELECTs should return only one row and indexes can be used. The data and index pages are referenced often and probably stay in the buffer pool. The response time impact is thus minimal.

A *main storage table* is faster to access, at least until a certain number of rows in the table is reached. It is more complicated to maintain.

Control Table

Function name	Program	Transaction	Plan name	New TRANS ID
	P0	TRX0	PLAN0	*
Sales	P1	TRX0	PLAN0	
Order	P2	TRX0	PLAN0	
Pay	P3	TRX0	PLAN0	
	PA	TRXA	PLANA	*
Price	PB	TRXA	PLANA	
Order	PC	TRXA	PLANA	
Parts	PD	TRXA	PLANA	
Price	PB	TEMP	PLANx	*

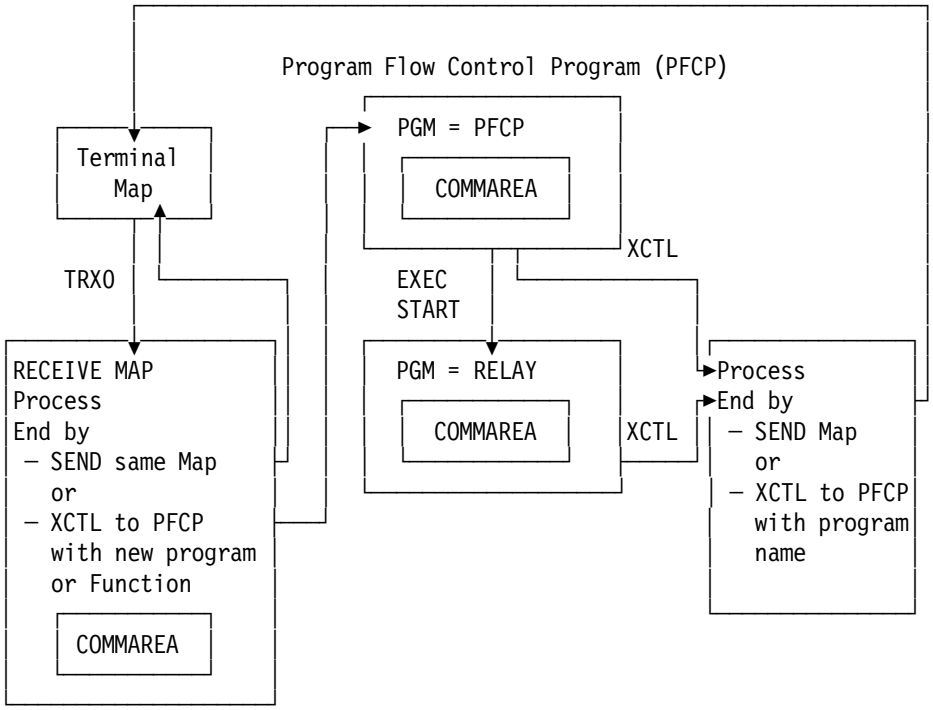


Figure 96. Table-Controlled Program Flow. The figure shows the principle of the table-controlled program flow technique.

10.3.7.2 Program Flow

The principle in the program flow is shown in Figure 96 on page 208. The flow for the application design used in Figure 95 on page 201 is explained below:

1. The terminal user sends a transaction to CICS. The transaction ID is TRX0.
2. The transaction definition points to program P0.
3. Program P0 receives the map, does some processing, and decides that program P3 is needed.
4. Instead of transferring control to program P3 (the DBRM for P3 could be part of another plan), P0 transfers control to the program flow control program (called PFCP in this example). P0 also passes a COMMAREA.
5. PFCP does a table lookup in the control table to see if program P3 is included in the plan currently in use (PLAN0). This is done by checking if P3 is specified in the same row as the current transaction ID (TRX0). In the example, this is the case (line 4 in the table).
6. PFCP then transfers control to program P3. It also passes the COMMAREA it received from P0 to P3.
7. P3 processes the necessary SQL calls and finishes either by sending a map to the terminal or by transferring control to PFCP (not shown) to execute another program.
8. Assuming that this other program is PB, PFCP again checks whether PB is allowed to run under the current transaction ID, which still is TRX0.

   The table shows that PB must not be executed under TRX0. PFCP then examines the table to find a transaction ID under which program PB can be executed. In the example, both TRXA and TEMP are valid transaction IDs. However, TRXA is pointing to program PA in the transaction definition. The New_TRANS_ID column of the table shows that only the rows with an * can be used when searching for a new transaction ID to start a given program. In this case, it is the TEMP transaction.

   There are two possibilities for program names in the RDO transaction definition entry for the TEMP transaction ID:

   - The RDO transaction definition can point directly to the new program (PB). In this case, there must be a transaction ID for each program that could be started in this way. Also, to use the COMMAREA, the program being started must contain logic to find out whether it is being started by START or by gaining control from an XCTL.
   - The RDO transaction definition can point to a common program, here called the RELAY program. In this case, one or more transaction IDs can be used. All of them point to the RELAY program in the RDO transaction definition. The purpose of the RELAY is to transfer control to the desired program. All these programs are then never begun with START and do not need to handle this situation.

     The solution with the RELAY program is shown in Figure 96 on page 208.
9. PFCP starts the transaction TEMP, passing the COMMAREA.
10. The RELAY program is started. It must use an EXEC CICS RETRIEVE command to retrieve the COMMAREA.
11. From the COMMAREA, RELAY picks up the program name PB.
12. RELAY transfers control to PB, passing the COMMAREA.

13. The plan switch is completed.

10.3.7.3 Advantages

- This method allows you to implement different types of application design, such as using one large plan or many small plans.
- The decision of when to switch plans is taken away from the development process, and is not part of the coding.
- Only the control table needs to be updated when new programs are set into production. The existing programs do not need to be changed, even if they can call the new functions.
- The relationship between the transaction IDs, the DB2 plans, and the programs can be changed without changing the programs. However, the control table must then be changed.
- Information from the DB2 catalog (SYSPLAN and SYSDBRM) can be used to build the control table.
- Alternatively, the control table can be used to generate information about the DBRM structure of the plans.
- The control table contains information that can assist in defining the RCT, (if the plan name column is available).
- Other functions can be included in a control table structure, for example information about which transaction ID to use in the TRANSID option of the EXEC CICS RETURN command.

10.3.7.4 Disadvantages

The two major disadvantages of this technique are the costs of designing and developing the solution and the execution time overhead.

The cost of getting from program to program is approximately doubled. However, this should normally not correspond to more than a few percent increase in the processor time for the transaction. To decide whether or not to use such a solution, you should balance these disadvantages against the advantages.

10.3.8 Using Packages

As explained in section 10.2.6.3, “Implementation” on page 196, the use of dynamic plan switching is restricted to unprotected pool threads and requires an implicit or explicit CICS SYNCPOINT to allow switching between plans.

The use of packages removes not only the need for dynamic plan switching but also the need for SYNCPOINTs and the use of unprotected threads. In addition, the ability to handle a large number of packages in a plan containing a single package list entry for a collection provides an environment where protected threads can be used and thread reuse optimized.

For example, a transaction that uses dynamic plan switching currently could be converted to use packages as follows:

- Bind all of the DBRMs contained in the plan associated with the transaction into a single collection.
- Bind a new plan with a PKLIST containing a single wildcard entry for this collection.

- Modify the RCT entry for this transaction to use the new plan. Protected threads could now be used for this transaction to optimize thread reuse.

High-usage packages could be bound with RELEASE(DEALLOCATE), with low-usage packages bound with RELEASE(COMMIT). This would result in the high-usage packages being retained in the plan′s package directory until plan deallocation, while the low-usage packages would be removed from the directory, and the space in the EDM pool released.

The disadvantage of this approach is that the use of RELEASE(DEALLOCATE) requires intervention to force deallocation of highly-used packages allocated to long-running threads to allow a package to be rebound. Consider that to rebind a package is less expensive than to rebind a plan, in terms of the time spent for doing it.

DB2 2.3 had no accounting for individual packages. So, if you rely on accounting data for your operation you should consider setting up procedures when the information is collected by plan name. This restriction was removed in DB2 3.1, where you can have accounting at the package level.

10.3.8.1 Converting CICS Applications

As discussed previously, the use of packages solves the problems that dynamic plan switching addressed. Conversion of the plans for transactions that use dynamic plan switching to the use of packages allows greater flexibility in setting up the plans needed to support the CICS applications. The choices in the number of plans to be used could include:

- One plan for the application

  This approach gives greatest flexibility in defining the RCT for the application because the transactions involved can be grouped to better utilize protected threads and optimize thread reuse. The steps in converting to this environment are:

  1. Bind all DBRMs for the transactions in the application into packages using a single collection such as COLLAPP1.
  2. Bind a new plan, PLANAPP1, with a package list consisting of a single entry, COLLAPP1*.

     ```
     BIND PLAN (PLANAPP1) ..... PKLIST (COLLAPP1.*) ..
     ```

  3. In the RCT, replace the dynamic plan switching exit program name with the name of this new plan.
     For example, replace:

     ```
     DSNCRCT TYPE=ENTRY,THRDM=0,PLNPGME=DSNCUEXT,TXID=TRAN1,TWAIT=POOL, ...
     ```

     with

     ```
     DSNCRCT TYPE=ENTRY,THRDM=0,PLAN=PLANAPP1,TXID=TRAN1,TWAIT=POOL, ...
     ```

- One plan per transaction

  The steps in using this approach would include:

  1. Bind the DBRMs for groups of transactions or all transactions into packages. The collections to be used could be based on the transactions being converted, such a TRAN1, or on the application, as above. The latter approach is preferable because creating and maintaining collections on a transaction basis requires greater administrative effort particularly if there are many common routines.

2. Bind a new plan for each transaction with a package list referring to a single wildcard package list entry that would depend on the approach taken. Use TRAN1.* if the collections were based on transactions or COLLAPP1.* if a single collection was set up for all transactions:

```
BIND PLAN (TRAN1) .... PKLIST (TRAN1.*) ...
```

or

```
BIND PLAN (TRAN1) .... PKLIST (COLLAPP1.*) ...
```

3. Modify the RCT to replace the DPS exit with the plan names associated with the transactions and reassemble.

This approach is preferable when using DB2 V2.3 if accounting data by plan name is required. Using one plan for all transactions provides accounting data by transaction ID but not by the plan names previously associated with DPS.

A similar approach can be taken for converting all CICS applications whether DPS is used or not.

10.3.8.2 Dynamic Plan Switching with Packages

For dynamic plan switching users, if the value of a modifiable special register (for example, the CURRENT PACKAGESET register) is changed during processing and not reset to the initial state before the SYNCPOINT is taken, the following occur:

- The thread is not released by the CICS attachment facility.
- Thread reuse does not occur because the task continues to use this thread with the same plan allocated.
- Dynamic plan switching does not occur because the same thread and plan are used; that is, the dynamic plan switching exit is not taken on the first SQL statement issued following the SYNCPOINT.

Therefore you should take care to ensure that any modifiable special register is reset to its initial value before the SYNCPOINT is taken if you want to use dynamic plan switching.

10.4 Programming

This section describes a number of programming considerations.

10.4.1 SQL Language

The complete SQL language is available to the CICS programmer with only minor restrictions. For a detailed description on using the SQL language in a CICS program, see *IBM DATABASE 2 Application Programming and SQL Guide*.

In a CICS program, it is possible to use:

- Data manipulating language (DML)
- Data description language (DDL)
- GRANT and REVOKE statements.

CICS also supports both dynamic and static SQL statements.

However, for performance and concurrency reasons, we recommend that in general you do not issue DDL and GRANT and REVOKE statements in CICS. You should also limit dynamic SQL use.

The reason for these recommendations is that the DB2 catalog pages can be locked, with a lower concurrency level as a consequence. Also the resource consumption for these types of SQL statements is typically higher than resource consumption for static DML SQL statements.

10.4.2 Views

We generally recommend that you use views where appropriate. Some views, however, cannot be updated.

In a real-time, online system, you often need to update rows you have retrieved using views. If the view update restriction forces you to update the base table directly (or by using another view), you should consider only views that can be updated. In most cases this makes the program easier to read and modify.

10.4.3 Updating Index Columns

When updating columns that are used in one or more indexes, consider the following:

- When updating a field in a table, DB2 does not use any index containing this field to receive the rows. This includes the fields listed in the FOR UPDATE OF list in the DECLARE CURSOR statement. It is independent of whether the field is actually updated.
- A table space that today is nonpartitioned can be recreated with more than one partition. SQL updates are not allowed against a partitioning key field. That means that programs doing updates of these fields must be changed to use DELETE and INSERT statements instead.

10.4.4 Dependency of Unique Indexes

A programmer can take advantage of the fact that DB2 will only return one row from a table with a unique index, if the full key is supplied in the SELECT statement. A cursor is not needed in this case. However, if at a later time the index is changed and the uniqueness is dropped, then the program does not execute correctly when two or more rows are returned. The program then receives an SQL error code.

10.4.5 Commit Processing

CICS ignores any EXEC SQL COMMIT statement in your application programs. The DB2 commit must be synchronized with CICS, which means that your program must issue an EXEC CICS SYNCPOINT command. CICS then performs the commit processing with DB2. An implicit SYNCPOINT is always invoked by the EXEC CICS RETURN at EOT.

You should be aware of the actions taken at COMMIT:

- The LUW is completed. This means that all updates are committed in both CICS and in DB2.
- The thread is released for terminal-oriented transactions (unless a held cursor is open). If the thread is released and there is no use for it, it is terminated unless it is a protected thread.

- The thread is not released for non-terminal-oriented transactions. It first happens when the transaction is finished. See section 5.2.3, “Commit Processing” on page 78 for details.
- All opened cursors are closed.
- All page locks are released.
- If RELEASE(COMMIT) was specified in the BIND process:
  - Table space locks are released.
  - The cursor table segments of the plan in the EDM pool are released.
- Table space locks obtained by dynamic SQL are released independently of the BIND parameters.
- The thread can be terminated, depending on the RCT specifications.

10.4.6 Serializing Transactions

You may need to serialize the execution of one or more transactions. This typically occurs when the application logic was not designed to deal with concurrency and in cases where the risk of deadlocks is too high.

You should allow serialization only for low-volume transactions because of potential queueing time.

The following methods each have different serialization start and end times:

- *CICS transaction classes*. The CICS facility of letting only one transaction execute at a time in a CLASS is useful to serialize the complete transaction.
- *DB2 thread serialization*. In cases where the serialization may be limited to an interval from the first SQL call to commit (for terminal-oriented transactions), you can use your RCT specifications to ensure that only one thread of a specific type is created at one time. This technique allows concurrency for the first part of the transaction, and is useful if the first SQL call is not in the beginning of the transaction. Do not use this technique if your transaction updated other resources before it issues its first SQL statement.
- *CICS enqueue and dequeue*. If you know that the serialization period necessary is only a small part of the programs, then the CICS enqueue and dequeue technique can be useful. The advantage is that only the critical part of the transaction is serialized. This part can be as small as just one SQL statement. It allows a higher transaction rate than the other methods, because the serialization is kept to a minimum.

  The disadvantage compared to the other techniques is that the serialization is done in the application code and requires the programs to be changed.
- *LOCK TABLE statement*. We recommend that you do *not* use the LOCK TABLE statement.

  The LOCK TABLE statement can be used to serialize CICS transactions and other programs, if EXCLUSIVE mode is specified. Note that it is the whole table space that is locked, not the table referenced in the statement.

  The serialization starts when the LOCK statement is executed. The end time for the serialization is when the table space lock is released. This can be at commit point or at thread deallocation time.

Use this technique with care, because of the risk of locking the table space until thread deallocation time. However, this technique is the only one that works across the complete DB2 system. The other techniques are limited to controlling serialization of only CICS transactions.

10.4.7 Page Contention

When designing applications and databases, consider the impact of having many transactions accessing the same part of a table space. The words “hot spot” are often used to describe a small part of the table space, where the access density is significantly higher than the access density for the rest of the table space.

If the pages are used for SELECT processing only, there is no concurrency problem. The pages are likely to stay in the buffer pool; so little I/O activity takes place. However, if the pages are updated frequently, you may find that you have concurrency problems, because the pages are locked from first update until commit. Other transactions using the same pages have to wait. Deadlocks and timeouts often occur in connection with hot spots.

Two examples of hot spots are sequential number allocation and insert in sequence.

10.4.7.1 Sequential Number Allocation

If you use one or more counters to supply your application with new sequential numbers, consider the following:

- You should calculate the frequency of updates for each counter. You should also calculate the elapsed time for the update transaction, measured from update of the counter until commit. If the update frequency multiplied by the calculated elapsed time exceeds about 0.5 in peak hours, the queue time can be unacceptable.
- If you are considering having more than one counter in the same table space, you should calculate the total counter busy time.
- If the counters are placed in the same row, they are always locked together.
- If they are placed in different rows in the same table space, they can be in the same page. Since the locks are obtained at the page level, the rows are also locked together in this case.
- If the rows are forced to different pages of the same table space (for example by giving 99% free space) it is still possible that the transactions can be queued.

  When for example row 2 in page 2 is accessed, a table space scan can occur. The scan stops to wait at page number 1, if this page is locked by another transaction. You should therefore avoid a table space scan.
- If an index is defined to avoid the table space scan, it is uncertain whether it will be used. If the number of pages in the table space is low, the index will not be used.
- A solution is then to have only one counter in each table space. This solution is preferred, if more than one CICS system is accessing the counters.
- If only one CICS system is accessing the counters, a BDAM file can be an alternative solution. However, the possibility of splitting the CICS system

into two or more CICS systems at a later time can make this solution less attractive.

10.4.7.2 Insert in Sequence

In situations where many transactions are inserting rows in the same table space, you should consider the sequence of the inserted rows. If you base a clustering index on a field with a time stamp, or a sequential number, DB2 tries to insert all rows adjacent to each other. The pages where the rows are inserted can then be considered a hot spot.

Note that in the clustering index, all inserts are also in the same page, within a given period.

If there is more than one index and the nonclustering index is used for data retrieval, the risk of deadlock between index and data is increased. In general terms, the INSERT obtains the X-locks (exclusive locks) in the following order:

1. Clustering index leaf page
2. Data page
3. Nonclustering index leaf page.

When the SELECT statement uses the nonclustered index, the S-locks (shared locks) are obtained in this order:

1. Nonclustering index leaf page
2. Data page.

This is the opposite order to the order of the INSERT locks. Often the SELECT rate is higher for the new rows. This means that the data pages are common for the INSERT and the SELECT statements. Where the index page is also the same, a deadlock can occur.

A solution to the deadlock risk is to spread the rows by choosing another index as clustering.

The general methods of how to handle deadlock situations are described in section 7.3, “Handling Deadlocks” on page 148.

Appendix A. Sample JCL and CICS Definitions

This appendix shows sample JCL and CICS resource definitions for CICS/ESA 3.3 with DB2 3.1, and CICS/ESA 4.1 with DB2 3.1. Similar examples are given in each chapter. However, they are grouped here for your convenience. Explanations of the meaning of JCL and parameters are extensively covered in each individual chapter.

A.1 CICS Procedures

```
//CICTAB33 PROC
//ASSEM   EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DUMMY
//SYSUT2   DD  DSN=&&TEMPPDS(MACROS),DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//             SPACE=(TRK,(1,1,5))
//*
//ASM     EXEC PGM=IEV90,REGION=4096K,COND=(3,LT,ASSEM),
//             PARM='SYSPARM(INITIAL),DECK,NOOBJECT,ALIGN'
//SYSLIB   DD  DSN=CICS330.SDFHMAC,DISP=SHR
//         DD  DSN=CICS330.SDFHSAMP,DISP=SHR
//         DD  DSN=SYS1.MODGEN,DISP=SHR
//         DD  DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSPUNCH DD  DSN=&&OBJMOD,DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),
//             SPACE=(400,(100,100))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DSN=&&TEMPPDS(MACROS),DISP=(OLD,PASS)
//*
//BLDMBR  EXEC PGM=IEBUPDTE,PARM=NEW,COND=((3,LT,ASSEM),(7,LT,ASM))
//SYSPRINT DD  DUMMY
//SYSUT2   DD  DSN=&&TEMPPDS,DISP=(OLD,PASS)
//SYSIN    DD  DSN=&&OBJMOD,DISP=(OLD,DELETE)
//*
//LNKEDT  EXEC PGM=IEWL,COND=((3,LT,ASSEM),(7,LT,ASM),(3,LT,BLDMBR)),
//             PARM='NORENT,LIST,XREF,LET,NCAL'
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1024,(100,50))
//SYSPRINT DD  SYSOUT=*
//SYSLMOD  DD  DSN=USER.CICS.TESTLIB,DISP=OLD
//SYSPUNCH DD  DSN=&&TEMPPDS,DISP=(OLD,PASS)
//SYSLIN   DD  DSN=&&TEMPPDS(LNKCTL),
//             DISP=(OLD,PASS),VOL=REF=*.ASSEM.SYSUT2
```

Figure 97. Procedure to Assemble a CICS/ESA 3.3 Table

```
//CICTAB41 PROC
//ASSEM   EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DUMMY
//SYSUT2   DD  DSN=&&TEMPPDS(MACROS),DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),
//             SPACE=(400,(200,100,5))
//*
//ASM     EXEC PGM=IEV90,REGION=4096K,COND=(3,LT,ASSEM),
//             PARM='SYSPARM(INITIAL),DECK,NOOBJECT,ALIGN'
//SYSLIB   DD  DSN=CICS410.SDFHMAC,DISP=SHR
//         DD  DSN=CICS410.SDFHSAMP,DISP=SHR
//         DD  DSN=SYS1.MODGEN,DISP=SHR
//         DD  DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSPUNCH DD  DSN=&&OBJMOD,DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),
//             SPACE=(400,(100,100))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DSN=&&TEMPPDS(MACROS),DISP=(OLD,PASS)
//*
//BLDMBR  EXEC PGM=IEBUPDTE,PARM=NEW,COND=((3,LT,ASSEM),(7,LT,ASM))
//SYSPRINT DD  DUMMY
//SYSUT2   DD  DSN=&&TEMPPDS,DISP=(OLD,PASS)
//SYSIN    DD  DSN=&&OBJMOD,DISP=(OLD,DELETE)
//*
//LNKEDT  EXEC PGM=IEWL,COND=((3,LT,ASSEM),(7,LT,ASM),(3,LT,BLDMBR)),
//             PARM='NORENT,LIST,XREF,LET,NCAL'
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1024,(100,50))
//SYSPRINT DD  SYSOUT=*
//SYSLMOD  DD  DSN=USER.CICS.V4R1.TESTLIB,DISP=SHR
//SYSPUNCH DD  DSN=&&TEMPPDS,DISP=(OLD,PASS)
//SYSLIN   DD  DSN=&&TEMPPDS(LNKCTL),
//             DISP=(OLD,PASS),VOL=REF=*.ASSEM.SYSUT2
```

Figure 98. Procedure to Assemble a CICS/ESA 4.1 Table

```
//DFHAUPLE PROC INDEX='CICS330',USER='USER.CICS',
//         DB2IND='DB2.V3R1'  for DB2 V3.1
//         DB2IND='DB2.V2R3'  for DB2 V2.3
//*
//ASSEM   EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DUMMY
//SYSUT2   DD  DSN=&&TEMPPDS(MACROS),UNIT=SYSDA,DISP=(,PASS),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),
//             SPACE=(400,(200,100,5))
//*
//ASM     EXEC PGM=IEV90,REGION=4096K,COND=(3,LT,ASSEM),
//             PARM='SYSPARM(INITIAL),DECK,NOOBJECT,FLAG(0)'
//SYSLIB   DD  DSN=&INDEX..SDFHMAC,DISP=SHR
//    DD  DSN=&DB2IND..SDSNMACS,DISP=SHR for DB2 V3.1
//    DD  DSN=&DB2IND..DSNMACS,DISP=SHR for DB2 v2.3
//         DD  DSN=SYS1.MACLIB,DISP=SHR
//         DD  DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSPUNCH DD  DSN=&&OBJMOD,DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),
//             SPACE=(400,(100,100))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DSN=&&TEMPPDS(MACROS),DISP=(OLD,PASS)
//*
//BLDMBR  EXEC PGM=IEBUPDTE,PARM=NEW,
//             COND=((3,LT,ASSEM),(7,LT,ASM))
//SYSPRINT DD  DUMMY
//SYSUT2   DD  DSN=&&TEMPPDS,DISP=(OLD,PASS)
//SYSIN    DD  DSN=&&OBJMOD,DISP=(OLD,DELETE)
//*
//LNKEDT  EXEC PGM=IEWL,PARM='LIST,XREF,LET,NCAL',
//             COND=((3,LT,ASSEM),(7,LT,ASM),(3,LT,BLDMBR))
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1024,(100,50))
//SYSPRINT DD  SYSOUT=*
//SYSLMOD  DD  DSN=&USER..TESTLIB,DISP=SHR
//SYSPUNCH DD  DSN=&&TEMPPDS,DISP=(OLD,PASS)
//SYSLIN   DD  DSN=&&TEMPPDS(LNKCTL),
//             DISP=(OLD,PASS),VOL=REF=*.ASSEM.SYSUT2
//*
```

Figure 99. Procedure to Assemble a Resource Control Table

A.2 CICS Program List Tables

```
//PLTPI  JOB  MSGCLASS=H,CLASS=B,MSGLEVEL=(1,1),REGION=4096K
//*
//PLEASE EXEC CICTAB33
//ASSEM.SYSUT1 DD *
       PRINT ON,NOGEN
         DFHPLT  TYPE=INITIAL,SUFFIX=3I
                 .
                 .
         DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
    DFHPLT TYPE=ENTRY,PROGRAM=DSNCCOM1
    DFHPLT TYPE=ENTRY,PROGRAM=DSNCCOM0 if PN29087 or DB2 3.1
                 .
                 .
Your initialization programs
                 .
                 .
         DFHPLT TYPE=FINAL
         END   DFHPLTBA
//*
//PLEASE EXEC CICTAB33
//ASSEM.SYSUT1 DD *
       PRINT ON,NOGEN
         DFHPLT  TYPE=INITIAL,SUFFIX=3T
                 .
                 .
  Your termination programs
                 .
                 .
    DFHPLT TYPE=ENTRY,PROGRAM=DSNCCOM2
         DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
                 .
                 .
         DFHPLT TYPE=FINAL
         END
```

Figure 100. CICS/ESA 3.3 Program List Tables Sample Job

```
//PLTPI  JOB  MSGCLASS=H,CLASS=B,MSGLEVEL=(1,1),REGION=4096K
//*
//PLEASE EXEC CICTAB41
//ASSEM.SYSUT1 DD *
       PRINT ON,NOGEN
         DFHPLT  TYPE=INITIAL,SUFFIX=4I
                .
                .
         DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
    DFHPLT TYPE=ENTRY,PROGRAM=DSN2COM0
                .
                .
Your initialization programs
  These programs could have
      SQL statements.
                .
                .
         DFHPLT TYPE=FINAL
         END   DFHPLTBA
//*
//PLEASE EXEC CICSAB41
//ASSEM.SYSUT1 DD *
       PRINT ON,NOGEN
         DFHPLT  TYPE=INITIAL,SUFFIX=4T
                .
                .
  Your termination programs
                .
                .
         DFHPLT TYPE=ENTRY,PROGRAM=DSN2COM2
         DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
                .
                .
         DFHPLT TYPE=FINAL
         END
```

Figure 101. CICS/ESA 4.1 Program List Tables Sample Job

A.3 CICS System Initialization Tables

```
//CICSSIT  JOB 1,'CICS330 SIT33',
// MSGCLASS=H,CLASS=B,MSGLEVEL=(1,1),
// PASSWORD=SYSADM,USER=SYSADM,REGION=4096K
//PLEASE EXEC CICTAB33
//ASSEM.SYSUT1 DD *
DFHSIT33 DFHSIT TYPE=CSECT,                                          X
               AKPFREQ=200,            FREQUENCY OF ACTIVITY KEYPOINT X
               AMXT=31,                MAX ACTIVE TASKS              X
               AIEXIT=DFHZATDX,        AUTOINSTALL EXIT NAME         X
               AIQMAX=100,             AUTOINSTALL MAX QUEUE         X
               AIRDELAY=700,           AUTOINSTAL RESTART DELAY      X
               APPLID=CICS33H,         SPECIFIC,GENERIC              X
               BMS=FULL,               BASIC MAPPING SUPPORT         X
               CMXT=(20,20,20,20),     MAX TASK CLASS                X
               CICSSVC=220,            CICS SVC TYPE 3               X
               COBOL2=YES,             COBOL II                      X
               DSASZE=1600K,           DSA SIZE                      X
               DUMPDS=A,               FORCE DMPA AT STARTUP         X
               ENQPL=12,               ENQ CONTROL BLOCK SPACE - 4K  X
                  .                                                  X
                  .                                                  X
                  .                                                  X
                  .                                                  X
               PLTPI=3I,               PLT INITIALIZATION            X
               PLTSD=3T,               PLT TERMINATION               X
                  .                                                  X
                  .                                                  X
                  .                                                  X
               SUFFIX=33,                                            X
                  .                                                  X
               VTAM=YES,               VTAM ACCESS METHOD            X
               WRKAREA=512,            CWA SIZE 50 BYTES FOR CICSPARS X
               XDCT=NO,                                              X
               XFCT=NO,                                              X
               XJCT=NO,                                              X
               XPPT=NO,                                              X
               XTRAN=NO,                                             X
               XTST=NO,                                              X
               XRF=NO                  NO XRF FOR BASIC LOCAL PROC
         END  DFHSITBA
```

Figure 102. CICS/ESA 3.3 System Initialization Table Sample Job

```
//CSIT41   JOB 1,'CICS410 SIT41',MSGCLASS=H,CLASS=A,MSGLEVEL=(1,1),
// PASSWORD=SYSADM,USER=SYSADM,REGION=4096K
//PLEASE EXEC CICTAB41
//ASSEM.SYSUT1 DD *
DFHSIT41 DFHSIT TYPE=CSECT,                                          X
               AIEXIT=DFHZATDX,         AUTOINSTALL EXIT NAME        X
               AILDELAY=0,                                           X
               AIQMAX=100,              AUTOINSTALL MAX QUEUE        X
               AIRDELAY=700,            AUTOINSTAL RESTART DELAY     X
               AKPFREQ=200,             FREQUENCY OF ACTIVITY KEYPOINT X
               APPLID=CICS410,          SPECIFIC,GENERIC             X
               BMS=FULL,                ALLOW PAGING, ETC            X
               CICSSVC=220,             CICS SVC TYPE 3              X
               CLSDSTP=NOTIFY,                                       X
               CLT=B0,                                               X
               CSDLSRNO=1,                                           X
               CSDJID=NO,                                            X
               DBP=1$,                  DYNAMIC BACKOUT PROGRAM      X
               DBUFSZ=2048,             DYNAMIC BUFFER               X
               DCT=41,                  EXTRAPARTITION TO SYSOUT     X
               DFLTUSER=SYSADM,                                      X
               INITPARM=(DSN2STRT='41,DB2A')  RCT and DB2            X
                  .                                                  X
                  .                                                  X
                  .                                                  X
                  .                                                  X
               PLTPI=4I,                PLT INITIALIZATION           X
               PLTSD=4T,                PLT TERMINATION              X
                  .                                                  X
               SUFFIX=41,                                            X
                  .                                                  X
                  .                                                  X
                  .                                                  X
               XRF=NO,                                               X
               VTAM=YES                 VTAM ACCESS METHOD
         END  DFHSITBA
```

Figure 103. CICS 4.1 System Initialization Table Sample Job

A.4 Resource Control Tables

```
//RCTASM3D JOB USER=SYSADM,PASSWORD=SYSADM,MSGCLASS=Z,CLASS=A,
//             MSGLEVEL=(1,1),REGION=4096K
//*********************************************************************
//*  THIS JOB USES THE PROCEDURE DFHAUPLE AT THE BEGINNIG TO          *
//*       ASSEMBLE THE RCT                                            *
//*********************************************************************
//RCT     EXEC DFHAUPLE,DB2IND='DSN310 or DSN230',
//        INDEX='CICS330'
//************************************************************
//* CICS 3.3 RCT
//************************************************************
//*
//ASSEM.SYSUT1   DD *
          DSNCRCT TYPE=INIT,SUBID=DB2A,STRTWT=YES,SNAP=A,             X
               ERRDEST=(CSMT,*,*),TOKENI=NO,SIGNID=SYSADM,            X
               SUFFIX=3,THRDMAX=9,SHDDEST=CSMT,TWAITI=NO,DPMODI=LOW
*--------------------------------------------------------------------*
*       COMMAND DEFINITION                                           |
*--------------------------------------------------------------------*
         DSNCRCT TYPE=COMD,DPMODE=HIGH,TXID=DSNC,                    X
               AUTH=(USERID,*,*),                                    X
               ROLBE=NO,                                             X
               TWAIT=POOL
*--------------------------------------------------------------------*
*       POOL DEFINITION                                              |
*--------------------------------------------------------------------*
         DSNCRCT TYPE=POOL,                                          X
               AUTH=(USERID,TXID,*),                                 X
               THRDM=3,THRDA=3,THRDS=3,                              X
               TWAIT=YES
*--------------------------------------------------------------------*
*       DB2 SAMPLE APPLICATION                                       |
*--------------------------------------------------------------------*
         DSNCRCT TYPE=ENTRY,TXID=D8PS,THRDM=1,THRDA=1,PLAN=DSN8CP23
         DSNCRCT TYPE=ENTRY,TXID=D8PP,THRDM=1,THRDA=1,PLAN=DSN8CQ23
         DSNCRCT TYPE=ENTRY,TXID=D8CS,THRDM=1,THRDA=1,PLAN=DSN8CC23
         DSNCRCT TYPE=ENTRY,TXID=D8PT,THRDM=1,THRDA=1,PLAN=DSN8CH23
         DSNCRCT TYPE=ENTRY,TXID=D8PU,THRDM=1,THRDA=1,PLAN=DSN8CH23
                  .
                  .
         YOUR RCT ENTRIES
                  .

         DSNCRCT TYPE=FINAL
         END
//
```

Figure 104. CICS/ESA 3.3 Resource Control Table Sample Job

```
//RCTASM41 JOB  USER=SYSADM,PASSWORD=SYSADM,MSGCLASS=Z,CLASS=A,
//             MSGLEVEL=(1,1),REGION=4096K
//*************************************************************
//* CICS 4.1 RCT
//*************************************************************
//*
//PLEASE EXEC CICTAB41
//ASSEM.SYSUT1 DD *
       PRINT ON,GEN
          DSNCRCT TYPE=INIT,SUBID=SSOP,STRTWT=YES,SNAP=A,              X
               ERRDEST=(CSMT,****,****),TOKENI=YES,PCTEROP=AEY9,       X
               SUFFIX=41,THRDMAX=30,SHDDEST=CSMT,TWAITI=NO,DPMODI=LOW
*-------------------------------------------------------------------*
*       COMMAND DEFINITION                                          |
*-------------------------------------------------------------------*
          DSNCRCT TYPE=COMD,DPMODE=HIGH,TXID=DSNC,                    X
               AUTH=(USERID,*,*),                                     X
               ROLBE=NO,                                              X
               TWAIT=POOL
*-------------------------------------------------------------------*
*       POOL DEFINITION                                             |
*-------------------------------------------------------------------*
          DSNCRCT TYPE=POOL,TXID=(PP01,PP02),                         X
               AUTH=(SYSADM,*,*),                                     X
               THRDM=3,THRDA=3,THRDS=0,                               X
               PLNEXIT=YES,PLNPGME=DSNCUEXT,                          X
               TWAIT=YES
*-------------------------------------------------------------------*
*       DB2 SAMPLE APPLICATION                                      |
*-------------------------------------------------------------------*
          DSNCRCT TYPE=ENTRY,TXID=D8PS,THRDM=1,THRDA=1,PLAN=DSN8CP23
          DSNCRCT TYPE=ENTRY,TXID=D8PP,THRDM=1,THRDA=1,PLAN=DSN8CQ23
          DSNCRCT TYPE=ENTRY,TXID=D8CS,THRDM=1,THRDA=1,PLAN=DSN8CC23
          DSNCRCT TYPE=ENTRY,TXID=D8PT,THRDM=1,THRDA=1,PLAN=DSN8CH23
          DSNCRCT TYPE=ENTRY,TXID=D8PU,THRDM=1,THRDA=1,PLAN=DSN8CH23
                   .
                   .
          YOUR RCT ENTRIES
                   .
                   .

*
          DSNCRCT TYPE=FINAL
          END
```

Figure 105. CICS/ESA 4.1 Resource Control Table Sample Job

A.5 CICS Startup JCL

```
//CICS33A  JOB  CLASS=K,USER=SYSADM,PASSWORD=SYSADM,
//              MSGLEVEL=(1,1),MSGCLASS=H,TIME=1440
//*--------------------------------------------------------------------
//ONLINE33 EXEC PGM=DFHSIP,TIME=1,REGION=6500K,DPRTY=(15,9),
//         PARM=('GRPLIST=(SNGLIST),DSNCRCT3,SYSIN')
//STEPLIB  DD  DSN=USER.CICS.TESTLIB,DISP=SHR,
//             UNIT=SYSDA,VOL=SER=USER01,DCB=BLKSIZE=32760
//         DD  DSN=USER.CICS.TESTLIB,DISP=SHR
//         DD  DSN=DSN310.SDSNLOAD,DISP=SHR
//         DD  DSN=CICS330.SDFHAUTH,DISP=SHR
//         DD  DSN=CICS330.SDFHLOAD,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2CICS,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2LIB,DISP=SHR
//         DD  DSN=PLI.V2R3M0.SIBMBASE,DISP=SHR
//         DD  DSN=PLI.V2R3M0.SIBMLINK,DISP=SHR
//         DD  DSN=PLI.V2R3M0.PLILINK,DISP=SHR
//DFHRPL   DD  DSN=CICS33D.ESA.SNTTABLE,DISP=SHR,
//             UNIT=SYSDA,VOL=SER=VOL001,DCB=BLKSIZE=32760
//         DD  DSN=USER.CICS.TESTLIB,DISP=SHR
//         DD  DSN=DSN310.SDSNLOAD,DISP=SHR
//         DD  DSN=CICS330.SDFHAUTH,DISP=SHR
//         DD  DSN=CICS330.SDFHLOAD,DISP=SHR
//         DD  DSN=CICS330.SDFHPL1,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2CICS,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2LIB,DISP=SHR
//         DD  DSN=PLI.V2R3M0.SIBMBASE,DISP=SHR
//         DD  DSN=PLI.V2R3M0.SIBMLINK,DISP=SHR
//         DD  DSN=PLI.V2R3M0.PLILINK,DISP=SHR
                .
                .
```

Figure 106. CICS/ESA 3.3 Sample STEPLIB and DFHRPL for Startup Job

```
//CICS41A  JOB CLASS=K,USER=SYSADM,PASSWORD=SYSADM,
//             MSGLEVEL=(1,1),MSGCLASS=H,TIME=1440
//*
//*--------------------------------------------------------------------*
//*        CICS START-UP                                               |
//*--------------------------------------------------------------------*
//ONLINE41 EXEC PGM=DFHSIP,TIME=1,REGION=0M,DPRTY=(15,9),
//         PARM=('SYSIN')
//STEPLIB  DD  DSN=USER.CICS.V4R1.TESTLIB,DISP=SHR
//         DD  DSN=CICS410.SDFHAUTH,DISP=SHR
//         DD  DSN=DB2.V3R1.SDSNLOAD,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2CICS,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2LIB,DISP=SHR
//         DD  DSN=SYS1.CSSLIB,DISP=SHR
//         DD  DSN=PLI.V2R3M0.SIBMBASE,DISP=SHR
//         DD  DSN=PLI.V2R3M0.SIBMLINK,DISP=SHR
//         DD  DSN=PLI.V2R3M0.PLILINK,DISP=SHR
//         DD  DSN=SYS1.PLIBASE,DISP=SHR
//DFHRPL   DD  DSN=USER.CICS.V4R1.TESTLIB,DISP=SHR
//         DD  DSN=CICS410.SDFHLOAD,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2CICS,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2LIB,DISP=SHR
//         DD  DSN=SYS1.CSSLIB,DISP=SHR
//         DD  DSN=PLI.V2R3M0.SIBMBASE,DISP=SHR
//         DD  DSN=PLI.V2R3M0.SIBMLINK,DISP=SHR
//         DD  DSN=PLI.V2R3M0.PLILINK,DISP=SHR
//         DD  DSN=SYS1.PLIBASE,DISP=SHR
                .
                .
           YOUR CICS JCL
                .
                .
//SYSIN    DD  *
*>>>  CICS SINGLE REGION SYSTEM  <<<*
APPLID=CICS41,
AUXTR=OFF,
FCT=NO,
GRPLIST=(C41LIST),
INITPARM=(DSN2STRT='41'),
PLTPI=4I,
PLTSD=4T,
SEC=YES,
SIT=41,
SKRPF7='P',    SINGLE PAGE RETRIEVAL - PREV PAGE
SKRPF8='N',    SINGLE PAGE RETRIEVAL - NEXT PAGE
START=COLD,
TCT=NO,
XCMD=NO,
XTRAN=NO
XPPT=NO,
XPCT=NO,
XFCT=NO,
XDCT=NO,
 .END
//
```

Figure 107. CICS/ESA 4.1 Sample STEPLIB and DFHRPL for Startup Job

A.6 CICS Precompile, Compile, and Bind Program Examples

```
//TESTCOB JOB  CLASS=A,REGION=4500K,
//             USER=SYSADM,PASSWORD=SYSADM,MSGCLASS=Z
//*********************************************************************
//*        DB2 PRE-COMPILER                                           |
//*********************************************************************
//DB2     EXEC PGM=DSNHPC,PARM='HOST(COBOL),XREF,SOURCE,FLAG(I),APOST'
//STEPLIB  DD  DSN=DSN310.SDSNLOAD,DISP=SHR
//         DD  DSN=USER.CICS.TESTLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSTERM  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//DBRMLIB  DD  DSN=USER.DBRMLIB(TESTCOB),DISP=SHR
//SYSCIN   DD  DSN=&&DBSOURCE,DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//             SPACE=(TRK,(10,1))
//SYSIN    DD  *
                 .
     YOUR CICS PROGRAM
                 .
//*********************************************************************
//*        CICS COMMAND LEVEL TRANSLATOR                              |
//*********************************************************************
//TRN     EXEC PGM=DFHECP1$,COND=(4,LT,DB2),
//             PARM='NOSOURCE,APOST,COBOL2,FLAG(I),VBREF,NOOPT'
//STEPLIB  DD  DSN=CICS330.SDFHLOAD,DISP=SHR
//SYSPUNCH DD  DSN=&&SOURCE,
//             DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//             SPACE=(TRK,(5,5))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DSN=&&DBSOURCE,DISP=(OLD,PASS)
//*********************************************************************
//*        COMPILE CICS PROGRAM                                       |
//*********************************************************************
//COB     EXEC PGM=IGYCRCTL,COND=((4,LT,DB2),(5,LT,TRN)),
// PARM='NOTRUNC,OBJECT,LIB,APOST,RES,RENT,LIST,DATA(31),NODYN,NOOPT'
//STEPLIB  DD  DSN=SYS1.COB2LIB,DISP=SHR
//SYSLIB   DD  DSN=SYS1.COB2CICS,DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT5   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSLIN   DD  DSN=&&OBJMOD(TESTC01),DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//             SPACE=(TRK,(5,2,1))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DSN=&&SOURCE,DISP=(OLD,DELETE)
```

Figure 108 (Part 1 of 2). DB2 3.1 and CICS/ESA 3.3 Example

```
//*********************************************************************
//*        LINKEDIT CICS PROGRAM                                      |
//*********************************************************************
//LNKEDT  EXEC PGM=IEWL,PARM='LIST,XREF,LET,AMODE=31,RMODE=31',
//             COND=((4,LT,DB2),(5,LT,TRN),(5,LT,COB))
//SYSLIB   DD  DSN=CICS330.SDFHLOAD,DISP=SHR
//         DD  DSN=SYS1.COB2CICS,DISP=SHR
//         DD  DSN=SYS1.COB2LIB,DISP=SHR
//         DD  DSN=DSN310.SDSNLOAD,DISP=SHR
//OBJLIB   DD  DSN=&&OBJMOD,DISP=(OLD,DELETE)
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT DD  SYSOUT=*
//SYSLMOD  DD  DSN=USRT001.DB2.V3R1.TESTLIB,DISP=SHR
//SYSLIN   DD  *
  INCLUDE SYSLIB(DFHECI)
  INCLUDE OBJLIB(TESTCOB)
  INCLUDE SYSLIB(DSNCLI)
  NAME TESTCOB(R)
//*********************************************************************
//*        BIND THIS PROGRAM                                          |
//*********************************************************************
//BIND    EXEC PGM=IKJEFT01,
//             COND=((4,LT,DB2),(5,LT,TRN),(5,LT,COB),(7,LT,LNKEDT))
//STEPLIB  DD  DSN=DSN310.SDSNLOAD,DISP=SHR
//DBRMLIB  DD  DISP=SHR,DSN=USER.DBRMLIB(TESTC01)
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
  DSN S(DB2A)
  BIND PLAN(TESTCOB)  MEMBER(TESTCOB) ACTION(REP) RETAIN -
  ISOLATION(CS)
END
//SYSIN    DD  DUMMY
//*********************************************************************
//*        GRANT PLAN EXECUTE                                         |
//*********************************************************************
//GRANT   EXEC PGM=IKJEFT01,COND=(4,LT,BIND)
//STEPLIB  DD  DSN=DSN310.SDSNLOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//DSNTRACE DD  SYSOUT=*
//SYSTSIN DD  *
  DSN S(DB2A)
  RUN PROGRAM(DSNTEP2) PLAN(DSNTEP31)
  END
//SYSIN    DD  *
  GRANT EXECUTE ON PLAN TESTCOB TO PUBLIC;
```

Figure 108 (Part 2 of 2). DB2 3.1 and CICS/ESA 3.3 Example

```
//TESTCOB4 JOB  CLASS=A,REGION=4500K, RESTART=BIND,
//             USER=SYSADM,PASSWORD=SYSADM,MSGCLASS=H
//*********************************************************************
//*        DB2 PRE-COMPILER                                           |
//*********************************************************************
//DB2     EXEC PGM=DSNHPC,PARM='HOST(COBOL),XREF,SOURCE,FLAG(I),APOST'
//STEPLIB  DD  DSN=DB2.V3R1.SDSNLOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSTERM  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//DBRMLIB  DD  DSN=USER.DBRMLIB(TESTCOB),DISP=SHR
//SYSCIN   DD  DSN=&&DBSOURCE,DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//             SPACE=(TRK,(10,1))
//SYSIN    DD  *
                .
     YOUR CICS PROGRAM
                .
//*********************************************************************
//*        CICS COMMAND LEVEL TRANSLATOR                              |
//*********************************************************************
//TRN     EXEC PGM=DFHECP1$,COND=(4,LT,DB2),
//             PARM='NOSOURCE,APOST,COBOL2,FLAG(I),VBREF,NOOPT'
//STEPLIB  DD  DSN=CICS410.SDFHLOAD,DISP=SHR
//SYSPUNCH DD  DSN=&&SOURCE,
//             DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//             SPACE=(TRK,(5,5))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DSN=&&DBSOURCE,DISP=(OLD,PASS)
//*********************************************************************
//*        COMPILE CICS PROGRAM                                       |
//*********************************************************************
//COB     EXEC PGM=IGYCRCTL,COND=((4,LT,DB2),(5,LT,TRN)),
//  PARM='LIB,APOST,RES,RENT,LIST,DATA(31),NODYN,NOOPT,MAP'
//STEPLIB  DD  DSN=SYS1.V1R3M2.COB2COMP,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2LIB,DISP=SHR
//SYSLIB   DD  DSN=USER.MACLIB,DISP=SHR  <-- map macro library
//         DD  DSN=CICS410.SDFHCOB,DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT5   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT6   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT7   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSLIN   DD  DSN=&&OBJMOD(TESTCO5),DISP=(,PASS),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//             SPACE=(TRK,(5,2,1))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DSN=&&SOURCE,DISP=(OLD,DELETE)
```

Figure 109 (Part 1 of 2). DB2 3.1 and CICS/ESA 4.1 Example

```
//*********************************************************************
//*        LINKEDIT CICS PROGRAM                                      |
//*********************************************************************
//LNKEDT  EXEC PGM=IEWL,PARM='LIST,XREF,LET,AMODE=31,RMODE=ANY',
//             COND=((4,LT,DB2),(5,LT,TRN),(5,LT,COB))
//SYSLIB   DD  DSN=CICS410.SDFHLOAD,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2CICS,DISP=SHR
//         DD  DSN=SYS1.V1R3M2.COB2LIB,DISP=SHR
//         DD  DSN=DB2.V3R1.SDSNLOAD,DISP=SHR
//OBJLIB   DD  DSN=&&OBJMOD,DISP=(OLD,DELETE)
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT DD  SYSOUT=*
//SYSLMOD  DD  DSN=USER.CICS.TESTLIB,DISP=SHR
//SYSLIN   DD  *
  INCLUDE SYSLIB(DFHECI)
  INCLUDE OBJLIB(TESTCOB)
  INCLUDE SYSLIB(DSNCLI)
  NAME TESTCOB(R)
//*********************************************************************
//*        BIND THIS PROGRAM                                          |
//*********************************************************************
//BIND    EXEC PGM=IKJEFT01,
//             COND=((4,LT,DB2),(5,LT,TRN),(5,LT,COB),(7,LT,LNKEDT))
//STEPLIB  DD  DSN=USER.CICS.TESTLIB,DISP=SHR
//         DD  DSN=DB2.V3R1.SDSNLOAD,DISP=SHR
//DBRMLIB  DD  DISP=SHR,DSN=USER.DBRMLIB(TESTCO5)
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
  DSN S(DB2A)
  BIND PLAN(TESTCOB) MEMBER(TESTCOB) ACTION(REP)  -
  RETAIN ISOLATION(CS) AQUIRE(USE)
END
//SYSIN    DD  DUMMY
//*********************************************************************
//*        GRANT PLAN EXECUTE                                         |
//*********************************************************************
//GRANT   EXEC PGM=IKJEFT01
//STEPLIB  DD  DSN=DB2.V3R1.SDSNLOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//DSNTRACE DD  SYSOUT=*
//SYSTSIN DD  *
  DSN S(DB2A)
  RUN PROGRAM(DSNTEP2) PLAN(DSNTEP31)
  END
//SYSIN    DD  *
  GRANT EXECUTE ON PLAN TESTCOB TO PUBLIC;
  COMMIT;
```

Figure 109 (Part 2 of 2). DB2 3.1 and CICS/ESA 4.1 Example

List of Abbreviations

ACEE	access control environment element
AOR	application-owning region
APF	authorized program facility
API	application program interface
BMP	batch message processing
BMS	basic mapping support
BSDS	bootstrap data set
CAF	call attachment facility
CCT	communication control table
CEC	central electronic complex
CICS	Customer Information Control System
CMF	CICS monitoring facility
CPU	central processing unit
CRC	command recognition character
CSD	CICS system definition file
CT	cursor table
DBRM	database request module
DBCTL	Database Control
DBD	data base descriptor
DDL	data description language
DML	data manipulating language
DPL	distributed program link
DTP	distributed transaction processing
EDF	execution diagnostic facility
EDM	environmental description manager
EOT	end-of-task
ERM	external resource manager
GTF	generalized trace facility
IBM	International Business Machines Corporation
IFCID	instrumentation facility component identifier
IFI	Instrumentation Facility Interface
ITSO	International Technical Support Organization
IRLM	IMS Resource Lock Manager
IVP	installation verification procedure
JCL	job control language
JCT	journal control table
LUW	logical unit of work
MASS	multiple address space subsystem
MQM	Message Queue Manager
MRO	multiregion operation
PCT	program control table
PI	program isolation
PLT	program list table
PLTPI	program list table for post-initialization
PLTSD	program list table for shutdown
PPT	processing program table
PT	package table
PTF	program temporary fix
RACF	Resource Access Control Facility
RCT	resource control table
RDO	resource definition online
RMF	Resource Measurement Facility
RMI	resource manager interface
SASS	single address space subsystem
SIT	system initialization table
SKCT	skeleton cursor table
SKPT	skeleton package table
SLR	Service Level Reporter
SMF	system monitoring facility
SMP	System Modification Program
SMP/E	System Modification Program Extended
SNT	sign-on table
SQL	Structured Query Language
SSI	Subsystem Interface
TCB	task control block
TPCP	two-phase commit protocol
TOR	terminal-owning region
TRUE	task related user exit
TS	table space
UR	unit of recovery
XRF	extended recovery facility

Index

E

F

G

H

I

L

T

U

V

X

ITSO Technical Bulletin Evaluation RED000

International Technical Support Organization
CICS/ESA-DB2 Interface Guide
September 1995

Publication No. SG24-4536-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction	____		
Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:
   Do you provide billable services for 20% or more of your time? Yes____ No____
   Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____
   If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

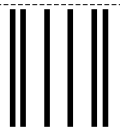
What other Technical Bulletins would you like to see published?

_____

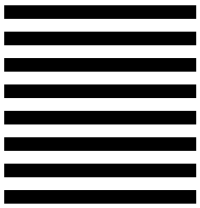
Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

Name	Address
Company or Organization	
Phone No.	

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department 471/E2
650 Harry Road
San Jose, CA
USA 95120-6099

Fold and Tape **Please do not staple** Fold and Tape

Printed in U.S.A.