CICS® Transaction Server for OS/390®

CICS DB2 Guide

Release 3

SC33-1939-34

CICS® Transaction Server for OS/390®

CICS DB2 Guide

Release 3

SC33-1939-34

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

Fourth edition (November 2000)

This edition applies to Release 3 of CICS Transaction Server for OS/390, program number 5655-147, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

This edition replaces and makes obsolete the previous edition, SC33-1939-33. Changes since that edition are indicated by a # sign to the left of a change. Any vertical lines in the left margin indicate a change made between Version 1 Release 2 and Version 1 Release 3 of CICS Transaction Server for OS/390.

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

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

IBM United Kingdom Laboratories, Information Development,
Mail Point 95, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

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

Contents

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Programming interface information

This book is intended to help you evaluate, install, and use the CICS DB2 Attachment Facility.

This book also documents Diagnosis, Modification or Tuning Information provided by CICS.

Diagnosis, Modification or Tuning Information is provided to help you diagnose problems with your CICS system.

Attention
Do not use this Diagnosis, Modification or Tuning Information as a programming interface.

Diagnosis, Modification or Tuning Information is identified where it occurs, by an introductory statement to a chapter or section.

This book contains sample programs. Permission is hereby granted to copy and store the sample programs into a data processing machine and to use the stored copies for study and instruction only. No permission is granted to use the sample programs for any other purpose.

Trademarks

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

CICS
CICS/ESA
CICS/MVS
CICSPlex SM
C/370
DATABASE 2
DB2
IBM
MVS/ESA
MVS Parallel Sysplex
NetView
OS/390
RACF
Resource Measurement Facility
RMF
VTAM
3090

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

Preface

Who this book is for

This book is for anyone who uses, or is considering using, the CICS Transaction Server for OS/390 Release 3 DB2 interface.

What this book is about

The aim of this book is to give introductory and guidance information on evaluating, installing, and using the CICS DB2 attachment facility.

This book is intended to be used in conjunction with existing manuals in the CICS and DB2 libraries. These are referred to where appropriate.

What you need to know before reading this book

Before you read this book, you need a general understanding of CICS. You can find general introductory information in the *CICS Family: General Information*. You should also have some knowledge of the concepts of data management and databases. For guidance on these topics, see the *DB2 for OS/390: Administration Guide*.

How to use this book

You will probably read this book sequentially. Aspects of DB2, from installation through performance considerations, are presented in the order in which you will need them.

Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both hardcopy and BookManager softcopy versions of a publication are usually in step. However, due to the time required to print and distribute hardcopy books, the BookManager version is more likely to have had last-minute changes made to it before publication.

Subsequent updates will probably be available in softcopy before they are available in hardcopy. This means that at any time from the availability of a release, softcopy versions should be regarded as the most up-to-date.

For CICS Transaction Server books, these softcopy updates appear regularly on the *Transaction Processing and Data Collection Kit* CD-ROM, SK2T-0730-xx. Each reissue of the collection kit is indicated by an updated order number suffix (the -xx part). For example, collection kit SK2T-0730-06 is more up-to-date than SK2T-0730-05. The collection kit is also clearly dated on the cover.

Updates to the softcopy are clearly marked by revision codes (usually a “#” character) to the left of the changes.

Notes on terminology

When the term “CICS” is used without any qualification in this book, it refers to the CICS element of IBM CICS Transaction Server for OS/390.

CICSPlex SM is used for CICSPlex System Manager.

“MVS” is used for the operating system, which can be either an element of OS/390, or MVS/Enterprise System Architecture System Product (MVS/ESA SP).

Bibliography

CICS Transaction Server for OS/390

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

CICS books for CICS Transaction Server for OS/390

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

CICSPlex SM books for CICS Transaction Server for OS/390

General	
CICSPlex SM Master Index	SC33-1812
CICSPlex SM Concepts and Planning	GC33-0786
CICSPlex SM User Interface Guide	SC33-0788
CICSPlex SM Web User Interface Guide	SC34-5403
CICSPlex SM View Commands Reference Summary	SX33-6099
Administration and Management	
CICSPlex SM Administration	SC34-5401
CICSPlex SM Operations Views Reference	SC33-0789
CICSPlex SM Monitor Views Reference	SC34-5402
CICSPlex SM Managing Workloads	SC33-1807
CICSPlex SM Managing Resource Usage	SC33-1808
CICSPlex SM Managing Business Applications	SC33-1809
Programming	
CICSPlex SM Application Programming Guide	SC34-5457
CICSPlex SM Application Programming Reference	SC34-5458
Diagnosis	
CICSPlex SM Resource Tables Reference	SC33-1220
CICSPlex SM Messages and Codes	GC33-0790
CICSPlex SM Problem Determination	GC33-0791

Other CICS books

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

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

Summary of Changes

This edition of the CICS DB2 Guide is based on the CICS DB2 Guide for CICS Transaction Server for OS/390 Release 2.

Major changes for this edition provide information on the changes to the INITPARM system initialization parameter now that CICS no longer supports running the CICS DB2 attachment facility by specifying the DSNCRCT macro in the PARM statement.

Information is provided on support for Java programs for CICS accessing DB2 (see "Support for Java™ programs" on page 72.) "Chapter 8. Customization" on page 77 explains the consequences of these changes for dynamic plan exits.

Information has been added on a safer way to purge CICS DB2 transactions using the DB2 CANCEL THREAD command (see "Purging CICS DB2 transactions" on page 19 and "Wait types" on page 121).

The changes in this book since the last edition are indicated by a vertical bar to the left of the text. Changes to the soft-copy release are indicated by the # symbol.

Chapter 1. Overview of the CICS Database 2 (DB2) interface

This chapter gives an overview of the CICS Transaction Server for OS/390 interface to Database 2 ™(DB2®).

This chapter includes the following sections:

Introduction to the CICS attachment facility

The CICS DB2 attachment facility provides the following major functions:

Application program interface
A CICS attachment facility is provided with CICS that allows you to operate DB2 with CICS. The connection between CICS and DB2 can be created or terminated at any time, and CICS and DB2 can be started and stopped independently. You also have the option of CICS automatically connecting and reconnecting to DB2. The DB2 system can be shared by several CICS systems, but each CICS system can be connected to only one DB2 subsystem at any one time.

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 are:
- STRT - start the connection to DB2
- STOP - stop the connection to DB2
- DISP - display the status of the connection to DB2
- MODI - modify characteristics of the connection to DB2
- DISC - disconnect threads

The CICS DB2 attachment facility provides a multithread connection to DB2. There is a thread for each active CICS transaction accessing DB2. Threads allow each CICS application transaction and DB2 command to access DB2 resources, such as a command processor or an application plan. A thread, used by a CICS transaction, or by a command, runs under its own subtask task control block (TCB). A thread based on the specifications in an RDO-defined definition is created when a transaction issues its first SQL statement. If an existing thread is available it is used before a new thread is created.

The CICS attachment facility provides the following thread types:

Command threads
Command threads are reserved by the CICS DB2 attachment facility for issuing commands to DB2 using the DSNC transaction. When the demand is great, commands overflow to the pool, and use a pool thread.

Entry threads
Entry threads are intended for high priority transactions. Unused protected entry threads are not terminated. Many CICS transactions can use the same protected entry thread. Entry threads defined as unprotected are terminated if no CICS transaction uses them.

Pool threads
Pool threads are used for all transactions and commands not using an entry

thread or a DB2 command thread. Pool threads are intended for low volume transactions, and overflow transactions from entry threads and DB2 command threads.

The CICS DB2 attachment facility provides CICS applications with access to DB2 data while operating in the CICS environment. CICS applications, therefore, can access both DB2 data and CICS data. CICS coordinates recovery of both DB2 and CICS data if transaction or system failure occurs. The relationship between DB2 and CICS is shown in Figure 1

DB2 requires several different address spaces as follows:

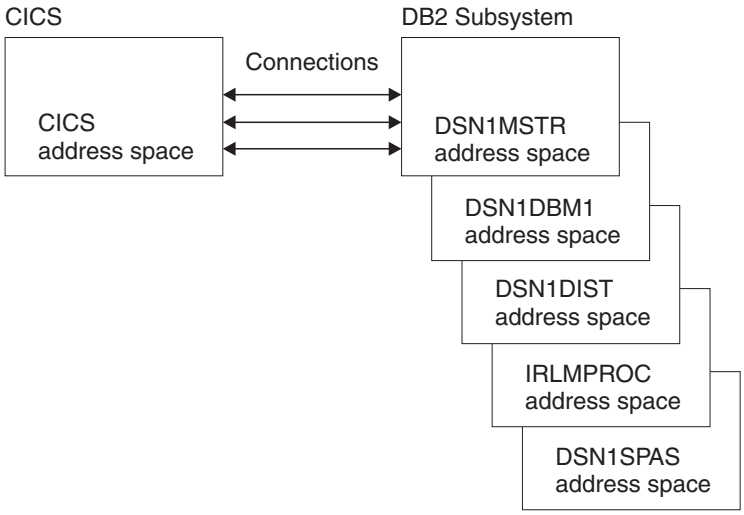


Figure 1. The relationship between DB2 and CICS

DSN1MSTR
for system services that perform a variety of system-related functions.

DSN1DBM1
for database services that manipulate most of the structures in user-created databases.

DSN1DIST
for distributed data facilities that provide support for remote requests.

IRLMPROC
for the internal resource lock manager (IRLM), which controls DB2 locking.

DSN1SPAS
for stored procedures, which provide an isolated execution environment for user-written SQL.

When an application program issues its first SQL request the CICS resource manager interface (RMI) processes the request and passes control to the attachment facility's task related user exit (TRUE). A transaction thread is scheduled by the CICS attachment facility, and at this stage DB2 checks authorization, creates control blocks and allocates the application plan. When the SQL request completes DB2 passes data back to the CICS attachment facility, and the CICS task regains control until the CICS transaction issues another SQL request. Finally, the CICS attachment facility returns control to the CICS application program.

Figure 2 on page 3 shows an overview of the CICS DB2 attachment facility.

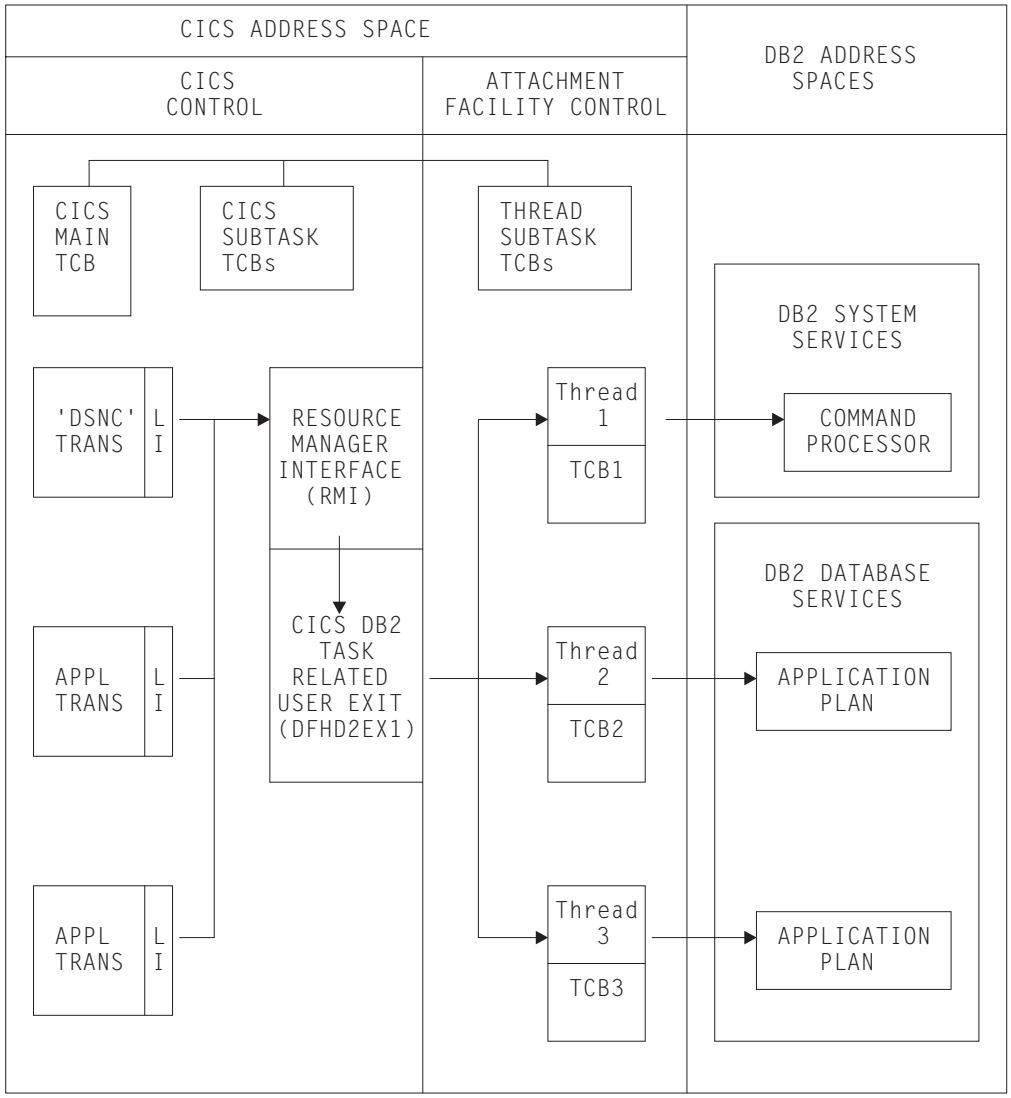


Figure 2. Overview of the CICS DB2 attachment facility

Overview of the CICS DB2 resource control table

In releases of CICS earlier than CICS Transaction Server for OS/390 Version 1 release 2, the connection between CICS and DB2 was defined in the resource control table (RCT). The RCT described the relationship between CICS transactions and DB2 resources (application plans and command processors) and was generated using the DSNCRCT macro provided by the CICS attachment facility.

CICS Transaction Server for OS/390 Release 3 no longer supports running the CICS DB2 attachment facility using the macro DSNCRCT. Instead, you must use an RCT that is defined using CICS resource definition online (RDO). Macro RCTs can still be assembled for the purpose of migrating them to the DFHCSD file only.

Using RDO the following objects can be defined and installed:

DB2CONN
Defines the global attributes of the CICS DB2 interface and defines the attributes of pool threads and command threads.

DB2ENTRY
Defines entry threads to be used by a specific transaction or group of transactions.

DB2TRAN
Defines additional transactions to be associated with a particular DB2ENTRY.

The objects can also be defined in batch using DFHCSDUP. CICSplex SM uses EXEC CICS CREATE to install these objects.

For more information about CICS DB2 definitions, see the *CICS Resource Definition Guide*.

Benefits of using online CICS DB2 resource definition

Using online CICS DB2 resource definition means that you do not have to shut down the interface between CICS and DB2 when adding, deleting, or changing CICS DB2 resource definitions. The benefits of using online definition for DB2 resources are discussed in the following sections:

- Function
- System availability
- Performance

Function

The function of the CICS DB2 attachment facility is enhanced by using online resource definition in the following ways:

- Existing macro-defined RCT load modules can be easily migrated to the CICS system definition file (CSD) using an RCT migration utility.
- The CICS DB2 attachment facility is fully integrated into CICS providing enhanced first failure data capture, enhanced messages with NLS support, more tracing including exception tracing, and system dump support with formatting integrated into the CICS IPCS verbexit.
- It enables management of the CICS DB2 interface by CICSplex SM, including single system image and single point of definition across a CICSplex.
- The CICS DB2 definitions become a CICS-managed resource which provides a consistent end user interface:
  - CEDA and DFHCSDUP for defining and installing entries
  - EXEC CICS CREATE for installing entries
  - CEMT/EXEC CICS INQUIRE, CEMT/EXEC CICS SET, and CEMT/EXEC CICS DISCARD for manipulating installed entries.

  Enhancements made to the CICS DB2 attachment facility before CICS Transaction Server for OS/390 Release 3 included:
  - Improved CICS DB2 statistics integrated with CICS
  - Improved attachment facility shutdown coordinated with CICS shutdown
  - Improved TCB management
  - Improved thread management and displays
  - A new accounting option

System availability

Online CICS DB2 resource definition allows you to add, delete or change definitions without the need to shut down the interface between CICS and DB2. You are therefore provided with continuous availability.

Performance

Online CICS DB2 resource definition provides benefits to performance as follows:

- CICS DB2 control blocks are moved above the 16MB line providing virtual storage constraint relief (VSCR).
- Online CICS DB2 resource definition provides the ability to specify generic transaction names, using wildcard symbols, which can reduce the number of definitions required in CICS.

Chapter 2. Installation and migration considerations

This chapter describes in the following sections how to migrate to the CICS Transaction Server Release 3 CICS DB2 attachment facility:

- "DB2 migration"
- "RDO for DB2 resource definitions" on page 8

The chapter summarizes the releases of IBM DATABASE 2 (DB2) supported by CICS, and the migration impact of RDO for CICS DB2 resource definitions.

DB2 migration

CICS supports the following releases of DB2:

- DB2 for MVS/ESA, Version 4 Release 1
- DATABASE 2 Server for OS/390 (DB2 for OS/930) Version 5 Release 1.
- DATABASE 2 Server for OS/390 (DB2 for OS/930) Version 6 Release 1.

CICS provides a CICS DB2 attachment facility (the CICS DB2 adaptor) that works with all supported releases of DB2. The CICS DB2 attachment facility is shipped on the CICS Transaction Server for OS/390 product tape, and you must use this version of the adaptor to connect a CICS Transaction Server region to DB2.

The CICS DB2 adaptor has been supplied by CICS since CICS/ESA® 4.1. Always use the correct CICS DB2 adaptor for the release of CICS under which a region is running—the CICS 4.1 adaptor for a CICS 4.1 region, and so on.

The DB2 program product continues to supply the earlier version of the CICS attachment facility to support DB2 connections with releases of CICS earlier than CICS 4.1.

The CICS DB2 adaptor and the DSNCRCT macro are now wholly owned by, and incorporated in, CICS. For information about this and other changes, see "RDO for DB2 resource definitions" on page 8.

To access DB2 databases using multisystem data sharing across an MVS Parallel Sysplex® requires DB2 4.1 or later.

Changes to the INITPARM system initialization parameter

| CICS no longer supports running the CICS DB2 attachment facility by specifying the DSNCRCT macro in the PARM statement, and a suffix should no longer be specified in the INITPARM parameter. The INITPARM parameter is now solely used to supply a DB2 subsystem override. The syntax of the INITPARM parameter is INITPARM=(DFHD2INI='*yyyy*') where *yyyy* is the (optional) 1–4 character DB2 subsystem identifier.

The DB2 subsystem identifier specified in the INITPARM system initialization parameter does not override a non-blank DB2 ID specified in the DB2CONN definition. For a description of the sequence in which the DB2 subsystem identifier is determined, see "CICS DB2 attachment operations with the CSD" on page 11.

CICS startup JCL requirements

The CICS DB2 attachment facility has to load the DB2 program request handler, DSNAPRH. To do this, the DB2 library, DSNxxx.SDSNLOAD, should be placed in the MVS linklist, or added to the STEPLIB concatenation of your CICS job.

To modify your CICS startup JCL you should concatenate the following libraries on the STEPLIB DD statement as follows:

- DSNxxx.SDSNLOAD (after the CICS libraries)

There should be no DB2 libraries in the DFHRPL DD statement. If DB2 libraries are required in the DFHRPL concatenation by an application, or by another product, they should be placed after the CICS libraries

RDO for DB2 resource definitions

You define DB2 resource definitions in the CSD as an alternative to assembling a DB2 resource control table. The DSNCRCT macro is shipped with CICS in order to allow migration of RCT tables to the CSD only. You should be aware of the changes to the macro described in the *CICS Transaction Server for OS/390 Migration Guide*.

All changes made to DB2 resource definitions installed directly from the CSD, or by using EXEC CICS CREATE commands, are cataloged and recovered in a CICS restart. Also, the DB2 objects installed from the CSD remain installed after the CICS DB2 attachment facility is stopped.

Effect on application programs

The removal of support for macro RCTs after creating the new type of DB2 resource definition means that application programs cannot access the RCT load module to obtain information about the connection, as in earlier releases of CICS.

Application programs running in earlier releases can find the address of the RCT by means of an EXEC CICS EXTRACT EXIT command that specifies the DB2 task-related user exit on the PROGRAM option, and specifies the GASET(*ptr_ref*) option to get the address of the global work area (GWA). The address of the RCT is stored by releases of the attachment facility earlier than CICS Transaction Server for OS/390 Release 2 at offset 8 in the GWA. Because the RCT is no longer available, the CICS DB2 attachment facility now sets offset 8 in the GWA to the address of fetch-protected storage, causing programs that use this address to fail with an ASRE abend. CICS issues message DFHSR0619 to the console and transient data queue CDB2 when an ASRE abend occurs.

You can use the DISMACP system initialization parameter to disable transactions that abend with an ASRE abend.

Migration using RDO for DB2 resource definition

You must migrate your existing RCT load modules into the CSD using the DFHCSDUP MIGRATE command. After migration, maintain the CSD definitions using RDO instead of maintaining the DSNCRCT macro definitions. To use the new CSD definitions, install the DB2 RDO definitions before the CICS DB2 attachment facility is actually started. You can modify dynamically most parameters of the DB2 resource definitions, and you can add new DB2ENTRY and DB2TRAN definitions while the CICS DB2 attachment facility is active.

To use the CICS DB2 attachment facility, the minimum migration requirements are:

1. **Reassemble your existing RCTs** with the DSNCRCT macro from CICSTS13.CICS.SDFHMAC.
2. **Migrate the reassembled RCTs** to the CSD using DFHCSDUP.
3. **Add the migrated GROUPs** to a suitable list specified on the GRPLIST system initialization parameter
4. **Change any INITPARM definitions** in the SIT or SIT overrides to remove any RCT suffix specified. The syntax of the INITPARM is INITPARM=(DFHD2INI='yyyy') (where yyyy is the 1– to 4 character DB2 subsystem ID). The INITPARM parameter is only used to override a blank DB2 ID in the DB2CONN definition. If the DB2CONN definition contains a non blank DB2 ID parameter this value is always used, regardless of the INITPARM specification.
5. **Upgrade the CSD** Run a DFHCSDUP job to upgrade the CICS DB2 definitions in the IBM-supplied group, DFHDB2. If you are migrating from CICS/ESA Version 3 or earlier, ensure that any previous user-defined group of CICS DB2 attach definitions are not installed — use the IBM-supplied group DFHDB2.
6. **Change the PLTPI** Change and reassemble the PLTPI table to specify program DFHD2CM0.

   This module replaces the DSN2COM0, DSN2COM1, DSNCCOM0, or DSNCCOM1 modules used in earlier releases to connect CICS to DB2 during startup.

   Alternatively, avoid specifying DFHD2CM0 in a PLTPI table altogether by specifying system initialization parameter DB2CONN=YES. This causes CICS to invoke the CICS DB2 attachment facility startup module automatically without requiring an entry in the PLTPI for the CICS DB2 adaptor module.
7. **Change the PLTSD table** to remove either DSN2COM0, DSN2COM1, DSNCCOM0, or DSNCCOM1, the modules used in earlier releases to disconnect CICS from DB2 during warm shutdown.

   The CICS DB2 attachment facility now uses the RMI shutdown function, meaning that it is called automatically to shutdown the interface when CICS is shutdown as follows:

   - The CICS DB2 interface is closed normally during a warm shutdown of CICS.
   - The CICS DB2 interface is force closed during an immediate shutdown of CICS.
   - Shutdown of the CICS DB2 interface is not initiated if CICS is cancelled or terminates abnormally.
8. **Change programs that START or STOP the CICS DB2 connection** Change application programs that start or stop the DB2 connection by linking to DSN2COM0, DSNCCOM0, DSN2COM1, or DSNCCOM1 (start) or DSN2COM1, DSNCCOM1, DSN2COM2, or DSNCCOM2, (stop).

   To start the CICS DB2 connection, change application programs to either:

   - Issue an EXEC CICS SET DB2CONN CONNECTED command (this is the recomended programming interface).
   - Link to DFHD2CM0 instead of DSN2COM0 or DSNCCOM0.

   To stop the CICS DB2 connection, change applications to either:

   - Issue an EXEC CICS SET DB2CONN NOTCONNECTED command (this is the recommended programming interface).
   - Link to DFHD2CM2 or DFHD2CM3 instead of DSN2COM2 or DSNCCOM2.

Check the use of EXTRACT EXIT PROGRAM(...) for DB2 interface
Applications issuing the EXEC CICS EXTRACT EXIT PROGRAM(...) ENTRYNAME(DSNCSQL) command, where the program name is DSNCEXT1 or DSN2EXT1, continue to work correctly. CICS dynamically substitutes program name DFHD2EX1, the CICS DB2 task-related user exit. Note that the entryname is still DSNCSQL. New application programs should use a program name of DFHD2EX1.

Similarly, application programs that issue the EXEC CICS INQUIRE EXITPROGRAM(DSN2EXT1) ENTRYNAME(DSNCSQL) command continue to work correctly. CICS automatically substitutes name DFHD2EX1. New application programs should use the exit program name DFHD2EX1.

In earlier releases of the CICS DB2 attachment facility, more than one task-related user exit is enabled. In addition to the exit with entryname DSNCSQL, earlier releases also support exits with entrynames DSNCIFC and DSNCCMD. Now, all requests through the CICS DB2 attachment facility are handled by a single task-related user exit program, DFHD2EX1, with entryname DSNCSQL. EXTRACT or INQUIRE EXITPROGRAM commands using entry names of DSNCIFC or DSNCCMD fail with INVEXITREQ (on the EXTRACT command) and PGMIDERR (on the INQUIRE command).

Migrating RCTs to the CSD

You can migrate existing RCTs to the CSD with the DFHCSDUP MIGRATE option using the following conventions:

- The utility migrates DSNCRCT TYPE=INIT, TYPE=POOL, and TYPE=COMD macro definitions to a DB2CONN definition with a default name of "RCT*xx*" where *xx* is the value of the SUFFIX= parameter.

  You can override the default DB2CONN name (RCT*xx*) by specifying your own name on the RDONAME parameter on the DSNCRCT TYPE=INIT macro definition. If RDONAME is specified on the TYPE=INIT macro, the DB2CONN is named by the RDONAME value.
- The utility migrates RCT TYPE=ENTRY to a DB2ENTRY definition named by the first non-generic transaction identifier on the TXID parameter.

  You can override the default DB2ENTRY name by specifying your own name on the RDONAME parameter on the TYPE=ENTRY macro definition. If RDONAME is specified on the TYPE=ENTRY, the DB2ENTRY is created with the RDONAME value.
- The utility creates a DB2TRAN definition that references the DB2ENTRY for each additional transaction identifier specified on the TXID parameter, using the TXID as the name of the DB2TRAN.

  You can override the default DB2TRAN name by specifying your own name on the RDONAME parameter on the TYPE=ENTRY macro definition. If the RDONAME is present on the TYPE=ENTRY and there is only one trans ID specified for that entry, the DB2TRAN is created with the RDONAME value.
- You can change the RCT source statements to define transaction IDs with generic names, using wildcard characters—the asterisk (*) and plus (+) symbols. However, CSD objects cannot have names that contain these generic symbols, so you must ensure that:
  - Each generic TXID is defined on a separate TYPE=ENTRY macro
  - The RDONAME parameter is specified to supply a valid CSD object name.

  For example, if you define:

```
DSNCRCT TYPE=ENTRY,TXID=J+++,RDONAME=J
DSNCRCT TYPE=ENTRY,TXID=D*,RDONAME=D
DSNCRCT TYPE=ENTRY,TXID=(ANDY,BILL,CARL),RDONAME=A
```

the following equivalent RDO definitions are generated:

```
DB2ENTRY(J)
DB2TRAN(J)    TRANSID(J+++) DB2ENTRY(J)
DB2ENTRY(D)
DB2TRAN(D)    TRANSID(D*)   DB2ENTRY(D)
DB2ENTRY(A)
DB2TRAN(ANDY) TRANSID(ANDY) DB2ENTRY(A)
DB2TRAN(BILL) TRANSID(BILL) DB2ENTRY(A)
DB2TRAN(CARL) TRANSID(CARL) DB2ENTRY(A)
```

You cannot define multiple generic transids referring to the same entry in the DSNCRCT macro, and these must be defined explicitly in the CSD using the DEFINE command.

Note that defining transaction IDs using wildcard characters removes the ability to collect CICS DB2 statistics on a per transaction basis as statistics are collected for each DB2ENTRY which will now represent a group of transactions.

- You can edit the DSNCRCT source macros to include a TYPE=GROUP, GROUP=*groupname*, to specify the group in which the utility should create the DB2 objects. All objects are created in the specified group until another TYPE=GROUP is encountered. If you omit the GROUPNAME parameter, DFHCSDUP uses a default group name of RCT*xx* where *xx* is the 1 or 2 character RCT suffix.

Changed defaults

The defaults for many of the new DB2 resource definition parameters are different from the defaults of their macro equivalents.

Most of the defaults of the DSNCRCT macro supplied in CICSTS13.CICS.SDFHMAC are unchanged from the DSNCRCT macro supplied in previous releases. If the DSNCRCT macro generates a default value that is changed from a previous release, it flags it with an MNOTE 5 during table assembly. The default parameters generated in an assembled RCT are migrated unchanged to the CSD.

CICS DB2 attachment operations with the CSD

Changes to the CICS DB2 attachment facility affect the operation of the facility in the following ways:

Attachment startup and shutdown

You can use the DSNC STRT command, but without a suffix, to start the CICS DB2 attachment facility.

The syntax of the DSNC STRT command is DSNC STRT *yyyy*. Any value specified after the STRT is treated as a DB2 subsystem override and not an RCT suffix.

CICS determines which DB2 subsystem ID to use from one of the following sources, in the order shown:

1. The DSNC STRT command, if specified
2. The DB2CONN resource definition, if DB2ID is specified

3. The INITPARM system initialization parameter, if specified (and DB2ID is blank in DB2CONN)
4. The default subsystem ID of DSN

You can use the DSNC STOP <QUIESCE|FORCE> command to stop the CICS DB2 attachment facility. The QUIESCE option now waits for all tasks to complete. In releases of CICS earlier than CICS Transaction Server for OS/390 1.2., a quiesce waits only for current units of work to complete.

During shutdown of the CICS DB2 attachment facility initiated by DSNC STOP, the terminal remains locked until the stop is complete, when message DFHDB2025 is issued.

As an alternative to the DSNC command, you can start and stop the CICS DB2 attachment facility using the EXEC CICS SET DB2CONN CONNECTED|NOTCONNECTED commands. You can also stop the CICS DB2 attachment facility by starting the CICS-supplied transactions CDBQ and CDBF from an application program, using an EXEC CICS START command. CDBQ causes a quiesce close and CDBF causes a force close.

CICS DB2 attachment facility command changes

The pool section of the DB2CONN resource definition does not have a TXID parameter associated with it. To modify the number of threads allowed on the pool, use reserved name CEPL on the DSNC MODIFY TRANS command. For example, issue the following command (where *n* is the new number of threads).

```
DSNC MODIFY TRANS CEPL n
```

The DSNC DISP TRAN *tttt* command now displays all threads running for a particular transid, instead of all the transactions associated with an RCT entry. In earlier releases, CICS uses the *tttt* operand to locate an RCT entry and then displays all the threads for that entry.

When you use the DSNC DISP STAT command, CICS displays statistics for DSNC commands on a line beginning `'*COMMAND'`. Pool thread statistics are displayed on a line beginning '*POOL'.

When modifying an RCT entry using the DSNC MODIFY TRANS *tttt* command, specify *tttt* exactly as it was defined in the RCT. If you defined a generic TXID, you must refer to the generic name when modifying it with a DSNC command. For example, if you have transactions called TAB and TAC, but they are defined generically as TA*, you can modify these on a DSNC command only by their generic name:

```
DSNC MODIFY TRANS TA*
```

and not by their specific names TAB and TAC.

SQL processing

User application programs do not need to be reassembled or rebound.

Dynamic plan exits

Dynamic plan exits can run unchanged and they do not need to be reassembled. The parameters passed to the exits are unchanged. However, you should be aware that dynamic plan switching can occur in new circumstances, and this could affect the operation of your dynamic plan exits.

A dynamic plan exit is invoked to determine which plan to use at the start of the first unit-of-work (UOW) of the transaction. This is referred to as *dynamic plan selection*.

A dynamic plan exit can also be invoked at the start of a subsequent UOW within the same transaction (provided the thread was released at syncpoint) to determine what plan to use for the next UOW. The plan exit can then decide to use a different plan. For this reason, this is referred to as *dynamic plan switching*.

In earlier releases of CICS, dynamic plan switching can occur only for the pool, or for RCT entries defined with THRDA (THREADLIMIT) specified as zero; that is, overflowed to the pool. A consequence of using DSNC MODIFY to modify THRDA to zero is that it makes dynamic plan switching effective. An RCT with THRDA greater than 0 is not capable of dynamic plan switching in earlier releases, and the plan selected for the first UOW is used for all subsequent UOWs of the transaction.

In CICS Transaction Server for OS/390 Release 3, dynamic plan switching can occur for both DB2ENTRYs as well as the pool, irrespective of the THREADLIMIT parameter. If you have coded your own dynamic plan exit, check that the logic can handle subsequent invocations for the same task. Your user application program, or the dynamic plan exit, must be written to tolerate consequences of additional calls to the exit. If the dynamic plan exit would change the plan when you don't want it to, the application program can avoid this by ensuring the thread is not released at syncpoint. However, the recommended method is to release the thread and ensure that the dynamic plan exit provides the correct plan for the new circumstances in which it is called.

Messages

The CICS message domain processes all attachment message requests. As a result, all previous DSNC messages now have the form DFHDB2*nnn*. You can use the DB2CONN MSGQUEUE1, MSGQUEUE2, and MSGQUEUE3 parameters to specify where the messages should be sent. This may have an impact on automation products, for example with NetView®.

Protected threads

If you use protected threads on DB2ENTRYs, note that a thread is no longer flagged as protected for its lifetime. Instead, a thread is protected only when it is not being used. If a new transaction reuses the thread, the thread is in use, and no longer requires protection. Therefore, the current number of protected threads for that DB2ENTRY is decremented. This allows for more effective protection of threads for a DB2ENTRY.

Chapter 3. Operations with CICS DB2

This chapter discusses operating the CICS DB2 attachment facility. It includes the following topics:

Starting the CICS DB2 attachment facility

The CICS DB2 attachment facility can be started automatically at initialization, or manually using the commands described in "Manual connection".

Automatic connection at CICS initialization

Connection between CICS and DB2 can be established automatically at CICS initialization by any one of the following methods:

- Specifying DB2CONN=YES in the SIT, or as a SIT override
- Specifying program DFHD2CM0 after the DFHDELIM statement of your PLTPI
- Specifying a user-written program that issues an EXEC CICS SET DB2CONN CONNECTED command, after the DFHDELIM statement of your PLTPI

Manual connection

Connection between CICS and DB2 can be established manually by any one of the following methods:

- Using the DSNC STRT command.

  For information on the DSNC STRT command see "DSNC STRT" on page 33.
- Using a CEMT SET DB2CONN CONNECTED command.

  For information on the SET DB2CONN CONNECTED command, see the *CICS Supplied Transactions* manual.
- Running a user application that issues an EXEC CICS SET DB2CONN CONNECTED command. For more information about the EXEC CICS SET DB2CONN CONNECTED command, see the *CICS System Programming Reference*

Stopping the CICS DB2 attachment facility

The CICS DB2 attachment facility, and hence the CICS-DB2 connection, can be quiesce stopped or force stopped. A quiesce stop waits for all CICS transactions currently using DB2 to complete. (In releases of CICS before CICS Transaction Server for OS/390 Release 2, a quiesce stop only waited for current units of work to complete.) A force stop causes all CICS transactions currently using DB2 to be forcepurged.

Automatic disconnection at CICS termination.

During startup of the CICS DB2 attachment facility, the CICS DB2 task-related user exit (TRUE) is enabled with the SHUTDOWN option. This means that CICS automatically invokes the TRUE when CICS is shut down, so as to shut down the CICS DB2 connection. When invoked during a quiesce shutdown of CICS, the CICS DB2 TRUE initiates a quiesce stop of the attachment facility. Likewise an immediate shutdown of CICS causes a force shutdown of the attachment facility to be initiated by the TRUE. If CICS is cancelled, no shutdown of the attachment facility is initiated by the TRUE.

There is no longer a need to specify a CICS DB2 program in PLTSD.

Manual disconnection

The connection between CICS and DB2 can be stopped or disconnected by using any one of the following methods:

- Using the DSNC STOP command.

  For information on the DSNC STOP command see “DSNC STOP” on page 31.
- Using a CEMT SET DB2CONN NOTCONNECTED command.
- Running the CICS-supplied CDBQ transaction that issues an EXEC CICS SET DB2CONN NOTCONNECTED command.

  The CDBQ transaction runs program DFHD2CM2. The transaction can be run directly from the terminal or by using the EXEC CICS START command. No messages are output to the terminal. The CICS DB2 adapter however outputs messages to transient data as part of its shutdown procedure.
- Running the CICS supplied CDBF transaction that issues an EXEC CICS SET DB2CONN NOTCONNECTED FORCE command.

  The CDBF transaction runs program DFHD2CM3. The transaction can be run directly from the terminal or EXEC CICS STARTed. No messages are output to the terminal. The CICS DB2 adapter, however, outputs messages to transient data as part of its shutdown procedure.
- Running a user application that issues an EXEC CICS SET DB2CONN NOTCONNECTED command.

Avoiding the need for manual resolution of indoubt units of work (UOWs)

Indoubt units of work (UOWs) can occur when CICS, DB2, or the whole system fails whilst a transaction is carrying out syncpoint processing, that is, during processing of an EXEC CICS SYNCPOINT or EXEC CICS RETURN command. CICS is the coordinator of the UOW, and if more than one recoverable resource is involved uses the two-phase commit protocol when trying to commit the UOW. Between replying ’yes’ to the phase 1 prepare call, and before receiving the commit or backout call at phase 2 time, DB2 is indoubt as to the outcome of the UOW. If a failure occurs at this time, DB2 remains indoubt about the UOW; it has an indoubt UOW and must ask CICS to resynchronize.

The indoubt UOWs are normally resolved automatically when the connection between CICS and DB2 is reestablished. CICS and DB2 exchange information regarding the indoubt UOWs, that is, CICS informs DB2 whether the UOW backed out or was committed. CICS maintains information about UOWs needed for resynchronization on its system log. Resynchronization information is maintained by CICS across warm, emergency, and cold starts. (Recovery of resynchronization

information across cold starts was introduced in CICS Transaction Server for OS/390 Version 1 release 1.) **Resynchronization information is lost if an initial start of CICS is performed as the system log is initialized and all information is lost, deleting all information about previous units of work.**

You should rarely need to initial start CICS. If you simply want to reinstall resources from the CSD, a cold start should be used, which allows any resynchronization information to be recovered. In particular, an initial start of CICS should be avoided if the previous warm shutdown of CICS issued message DFHRM0131 indicating that resynchronization is outstanding.

If CICS is initial started when DB2 resync was required, when the CICS DB2 connection is re-established, message DFHDB2001 is output for each UOW that failed to be resynchronized, and the UOW must be resynchronized in DB2 using the DB2 RECOVER INDOUBT command. CICS Transaction Server has no equivalent to the DFH$INDB utility that was available in CICS/ESA Version 4 and below, which allowed scanning of the system log to ascertain the outcome of the UOW. The MVS system log, and hence all the UOW information on it, has been lost by initial starting of CICS.

Monitoring the attachment facility

You can monitor and modify the status of the connection between CICS and DB2 using CEMT commands and commands provided by the CICS DB2 attachment facility. The same function provided by CEMT is also available via EXEC CICS INQUIRE and SET commands. Below is a summary of the functions available:

CEMT INQUIRE and SET DB2CONN
The global status of the connection and its attributes along with attributes of pool threads and command threads can be inquired upon and set using these commands.

CEMT INQUIRE and SET DB2ENTRY
The attributes of a particular DB2ENTRY defining threads to be be used by a specific transaction or set of transactions can be inquired upon and set using these commands.

CEMT INQUIRE and SET DB2TRAN
Use these commands to find out and alter which transaction IDs use which DB2ENTRYs.

DSNC DISC (disconnect)
This CICS DB2 attachment command can be used to cause currently connected threads to be terminated as soon as they are not being used by a transaction. For active threads, that means when the transaction releases the thread, which is typically at syncpoint time. Protected threads are threads that currently are not being used by a transaction and normally are released if they have not been reused after two protected thread purge cycles. A DSNC DISCONNECT commands pre empts the purge cycles and causes them to be terminated immediately. For more information on the DSNC DISCONNECT command see “DSNC DISCONNECT” on page 23.

DSNC DISP (display)
This CICS DB2 attachment command can be used to display the status of active threads for a particular plan, for a particular transaction, or for all plans and transactions. For more information, see “DSNC DISPLAY” on page 24.

The DSNC DISP command also displays CICS DB2 statistics to a CICS terminal. These statistics are those available in previous releases, but are only

a subset of the new CICS DB2 statistics available via CICS COLLECT STATISTICS and PERFORM STATISTICS commands. For more information on DSNC DISP STAT command, see “Parameter description” on page 25. For more information on the full CICS DB2 statistics available see the *CICS Performance Guide*.

DSNC MODI (modify)

This CICS DB2 attachment command can be used to modify where unsolicited messages are sent and the number of threads allowed for a DB2ENTRY or for the pool or command threads. This function is superseded by the CEMT commands described which allow these attributes to be modified and all other attributes of a DB2CONN, DB2ENTRY or DB2TRAN. For more information, see “DSNC MODIFY” on page 28.

Entering DB2 commands

Once the connection between CICS and DB2 has been established, terminal users authorized by CICS can use the DSNC transaction to route commands to the DB2 subsystem. These commands are defined as follows:

```
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) of ’-’ must be used to distinguish DB2 commands from CICS DB2 attachment facility commands. This command recognition character is not used to identify the DB2 subsystem to which the command is to be sent. The command is sent to the DB2 subsystem to which CICS is currently connected. Figure 3 shows the CICS attachment facility commands These require CICS authorization to use the DSNC transaction and the DB2 commands. For more information about the DSNC -DB2COMMAND, see “Issuing commands to DB2 using DSNC” on page 21.

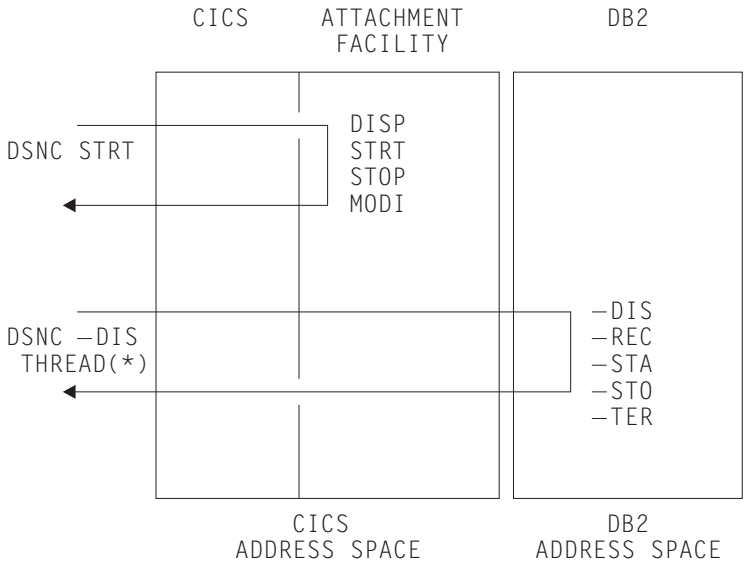


Figure 3. Examples of CICS attachment facility commands and some DB2 commands

Purging CICS DB2 transactions

CICS-DB2 applications that access DB2 will enter a CICS wait, with a resource type of 'DB2' and a resource name of 'LOT_ECB', whilst they wait for the CICS-DB2 subtask to access DB2 and complete the request. A CICS task in this wait cannot be purged, but forcepurge is supported. However, there is a risk when forcepurging a task in this state, in that it can cause the associated CICS-DB2 subtask to be terminated during a 'must complete' activity in DB2, which would cause termination of the DB2 subsystem.

DB2 'must complete' activities are short-lived, but to avoid this risk, a safer way to terminate a CICS-DB2 application, when using DB2 V4 or above, is to use the DB2 CANCEL THREAD command. If the CICS-DB2 subtask is active in DB2 at the time, then the thread is terminated, and 'must complete' activity is either avoided, or completed before termination. If the subtask is not active in DB2 at the time of the cancel, then the cancel is deferred until DB2 is next accessed using that thread. Once the DB2 CANCEL THREAD command has been issued, you can safely issue a forcepurge of the CICS transaction, if needed (because the task is currently active in CICS rather than in DB2).

To determine which DB2 thread is associated with a CICS task, use the DSNC DISPLAY TRAN command (see "DSNC DISPLAY" on page 24), which will show the CICS task number, transaction id, and the 12–byte DB2 correlation id of the associated DB2 thread used by the CICS-DB2 subtask. This correlation id uniquely identifies a thread. A DSNC —DIS THD(*), issues a DB2 display thread command showing all threads used by this CICS system identified by correlation id, and gives a unique token that can be used with a DB2 CANCEL THREAD command to cancel the thread.

Starting SMF for DB2 accounting, statistics and tuning

Through its instrumentation facility, DB2 produces a system management facility (SMF) type 101 accounting record at each CICS DB2 thread termination and at each CICS DB2 sign-on. DB2 also produces an SMF type 100 record to hold DB2 statistics on a DB2 sub system 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.

  To do this the MON,SMFACCT and/or SMFSTAT of the initialization macro DSN6SYSP are set to YES. See the *DB2 for OS/390: Administration Guide* more details.
- Activating SMF to write the records.

  Add entries for type 100, 101, and 102 records to the existing SMF member (SMFPRM*xx*) entry in SYS1.PARMLIB.
- Starting DB2 trace.

  Starting DB2 trace can be done at DB2 startup time by setting the parameters of the initialization macro DSN6SYSP, or by using the DB2 -START TRACE command giving the specific trace and classes to be started. The latter is the way to start recording accounting, statistics and performace data on a periodic basis.

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 (DB2PM) Program Product.

Starting GTF for DB2 accounting, statistics and tuning

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 which you can issue from a CICS terminal using the DSNC transaction. The commands can be used to specify:

Trace scope
: either GLOBAL for DB2 subsystem activity, PERF for performance, ACCTG for accounting, STAT for statistics, or MONITOR for monitoring activity.

Trace destination
: GTF, in main storage, or SMF.

Trace constraints
: specific plans, authorization IDs, classes, location and resource managers.

Chapter 4. CICS-supplied transactions

Information about the system programming function provided by CEMT for the following commands can be seen in the *CICS Supplied Transactions*. Information about the EXEC CICS INQUIRE and EXEC CICS SET commands is in the *CICS System Programming Reference* manual.

- INQUIRE DB2CONN
- SET DB2CONN
- INQUIRE DB2ENTRY
- SET DB2ENTRY
- INQUIRE DB2TRAN
- SET DB2TRAN

With the integration of the CICS DB2 attachment facility into CICS Transaction Server, the DSNC command interface is discussed in this chapter as a CICS-supplied transaction. This chapter contains the following sections:

CICS DB2 transaction DSNC

The DSNC transaction can be used to perform the following:

- Enter DB2 commands from a CICS terminal.
- Cause threads to be terminated when they are released (DSNC DISCONNECT).
- Display information about transactions using the CICS DB2 interface, and display statistics (DSNC DISPLAY).
- Modify the unsolicited message destinations, and modify the number of active threads used by a DB2ENTRY, the pool, or for commands (DSNC MODIFY).
- Shut down the CICS DB2 interface (DSNC STOP).
- Start the CICS DB2 interface (DSNC STRT).

Issuing commands to DB2 using DSNC

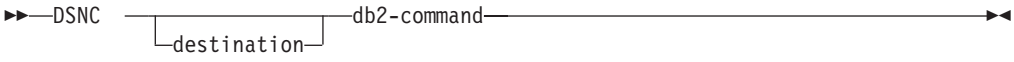
The CICS attachment facility DSNC command allows you to enter DB2 commands from CICS.

Environment

This command can be issued only from a CICS terminal

Syntax

DSNC syntax



Abbreviation

You can omit DSNC from the DB2 command if the relevant transaction definition is installed from the CICS DB2 sample group, DFH$DB2. For example, if you are installing the -DIS transaction definition.

```
DSNC -DIS THD(*)
```

can be abbreviated to:

```
-DIS THD(*)
```

The sample CICS DB2 group, DFH$DB2, contains the following transaction definitions for issuing DB2 commands:

-ALT	-ARC	-CAN
-DIS	-MOD	-REC
-RES	-SET	-STA
-STO	-TER	

Authorization

Issuing DB2 commands using DSNC does not require any authorization from CICS over and above transaction attach security required to run the DSNC transaction. It does, however, require DB2 privileges. For a description of the privileges required for each DB2 command, see “Authorizing commands to DB2” on page 60. For more information about CICS security, see “CICS security” on page 52.

Parameter description

destination
Identifies another terminal to receive display information. It must be a valid terminal that is defined to CICS and supported by basic mapping support (BMS).

db2-command
Specifies the exact DB2 command that you want to enter from a CICS terminal It must be preceded by a hyphen.

Usage note

Screen scrolling
The SIT keywords SKR*xxxx* can be used to support the scrolling of DSNC DB2 commands from your terminal. For further information about the SIT keywords and parameters, see the *CICS System Definition Guide*.

Example

Issue the DB2 command -DISPLAY THREAD from a CICS terminal to display threads for a CICS with applid IYK4Z2G1.

```
DSNC -DISPLAY THREAD(IYK4Z2G1)
```

```
DSNV401I : DISPLAY THREAD REPORT FOLLOWS -
DSNV402I : ACTIVE THREADS -
NAME     ST A   REQ ID           AUTHID   PLAN     ASID
IYK4Z2G1 N        3              JTILLI1           00BA
IYK4Z2G1 T        3 ENTRXC060001 CICSUSER TESTC06  00BA
IYK4Z2G1 T        3 POOLXP050002 CICSUSER TESTP05  00BA
IYK4Z2G1 T  *     6 COMDDSNC0003 JTILLI1           00BA
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I : DSNVDT '-DIS THREAD' NORMAL COMPLETION
DFHDB2301 07/09/98 13:36:36 IYK4Z2G1 DSNC DB2 command complete.
```

Figure 4. Sample output from DSNC -DISPLAY command

DSNC DISCONNECT

The CICS attachment facility command DSNC DISCONNECT disconnects threads.

The command provides manual control to release resources being shared by normal transactions so that special purpose processes, such as DB2 utilities, can have exclusive access to the resources.

Environment

This command can be issued only from a CICS terminal.

Syntax

DISC syntax

Abbreviation

DSNC DISC or DISC (using the DISC transaction from the CICS DB2 sample group DFH$DB2).

Authorization

Access to this command can be controlled using the following CICS authorization checks:

- Transaction attach security for transaction DSNC
- Command security for resource DB2CONN. This command requires READ access. For more information about CICS security, see “Chapter 2. Installation and migration considerations” on page 7.

Parameter description

plan-name
Specifies a valid application plan.

Usage notes

Preventing creation of threads

The command DSNC DISCONNECT does not prevent threads from being created on behalf of transactions. The command only causes currently connected threads to be terminated as soon as they are not being used by a transaction. To interrupt a transaction and cancel a thread faster, you can use the DB2 CANCEL THREAD command with DB2 Version 4 or higher.

You can stop the transactions associated with a particular plan ID in CICS with the MAXACTIVE setting for TRANCLASS. This prevents new instances of the transaction from causing a re-creation of a thread.

Alternative for protected threads

You may want to deallocate a plan for rebinding or for running a utility against the database. If you are using a protected thread use EXEC CICS SET DB2ENTRY(*entryname*) THREADLIMIT(0), or DSNC MODIFY rather than DSNC DISCONNECT, to send all the threads to the pool. The protected thread terminates on its own within two purge cycles. See the PURGECYCLE attribute of DB2CONN.

Example

Disconnect active and protected threads for plan TESTP05:

```
DSNC DISC TESTP05
```

```
DFHDB2021 07/09/98 13:46:29 IYK4Z2G1 The disconnect command is complete.
```

Figure 5. Sample output from DSNC -DISPLAY command

DSNC DISPLAY

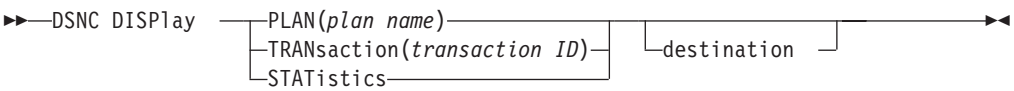
The CICS attachment facility command DSNC DISPLAY displays information on active CICS DB2 threads and the corresponding CICS transaction using them, or statistical information associated with DB2ENTRYs and the DB2CONN.

Environment

This command can be issued only from a CICS terminal.

Syntax

DISPLAY syntax



Abbreviation

DSNC DISP or DISP (using the DISP transaction from the CICS DB2 sample group DFH$DB2).

Authorization

Access to this command can be controlled using the following CICS authorization checks:

- Transaction attach security for transaction DSNC
- Command security for resource DB2CONN. This command requires READ access. For more information about CICS security, see “Chapter 2. Installation and migration considerations” on page 7.

Parameter description

plan-name
Displays information about threads by *plan name*. *Plan-name* is a valid plan name for which information is displayed.

If you do not specify *plan-name* (or if you specify an asterisk, *), information is displayed for all active threads.

Example
Display information on all active plan IDs listed in the resource control table. The display information is to be sent to another terminal designated as MTO2.

```
DSNC DISP PLAN * MTO2
```

Parameter description

transaction-ID
A valid transaction ID for which thread information is displayed.

If you do not specify a transaction ID, information is displayed for all active threads.

Example
Display information about all active transactions listed in the resource control table.

```
DSNC DISP TRAN
```

Parameter description

Displays the statistical counters associated with each entry in the resource control table. The counters concern the usage of the available connections of the CICS attachment facility to DB2.

Environment

If you issue this command from CICS while the CICS attachment facility is active but the DB2 sub system is not, a statistics display is produced with no obvious indication that the sub system is not operational. Message DFHDB2037 appears in the CICS message log to indicate that the attachment facility is waiting for DB2 to start.

Example
Display statistical counters associated with each entry in the resource control table.

```
DSNC DISP STAT
```

Note that a more detailed set of CICS DB2 statistics can be obtained using standard CICS statistics interfaces, for example, the commands EXEC CICS COLLECT STATISTICS and EXEC CICS PERFORM STATISTICS, or using the DFH0STAT sample program.

Alternative destination

destination

The identifier of another terminal to receive the requested display information. It must be a valid terminal that is defined to CICS and supported by basic mapping support (BMS).

Because the optional destination is sometimes preceded by an optional plan name or transaction ID in the command, each parameter must be unique and separately identifiable as either a name or a terminal identifier. If only one parameter is entered, it is first checked to see whether it is a plan name or a transaction ID, and it is then checked as a destination. To use a character string that is both a plan name or transaction ID and also a valid terminal identifier, you must use both the name and destination parameters to display the required information at the required terminal.

When an alternate destination is specified to receive the requested display information, the following message is sent to the requesting terminal:

```
DFHDB2032 date time applid alternate destination display command complete
```

DISPLAY PLAN or TRAN

Figure 6 shows an example of the output for the DSNC DISPLAY (PLAN or TRANSACTION) command. For each created thread the output shows the name of the DB2ENTRY or '*POOL' for the pool, or the '*COMMAND' for which it has been created.

```
DFHDB2013 07/09/98 15:26:47 IYK4Z2G1 Display report follows for threads
accessing DB2 DB3A
DB2ENTRY S PLAN     PRI-AUTH SEC-AUTH CORRELATION  TRAN TASK  UOW-ID
*POOL    A TESTC05  JTILLI1           POOLXC050001 XC05 01208 AEEEC0321ACDCE00
XC06     A TESTC06  JTILLI1           ENTRXC060003 XC06 01215 AEEEC0432F8EFE01
XP05     A TESTP05  JTILLI1           ENTRXP050002 XP05 01209 AEEEC03835230C00
XP05     I TESTP05  JTILLI1           ENTRXP050004
DFHDB2020 07/09/98 15:26:47 IYK4Z2G1 The display command is complete.
```

Figure 6. Sample output from DSNC DISPLAY (PLAN or TRANSACTION) command

A status of 'A' in the 'S' field indicates that the thread is active within a unit of work. A status of 'I' indicates a protected thread that is waiting for work.

The PLAN associated with the thread is displayed (there is no plan for command threads).

The PRI-AUTH field shows the primary authorization ID used for the thread. The SEC-AUTH field shows the secondary authorization ID (if any) for the thread.

The CORRELATION fields shows the 12-byte thread correlation ID which is made up as *eeeetttnnnn* where *eeee* is either COMD, POOL or ENTR indicating whether it is a command, pool or DB2ENTRY thread; *tttt* is the transid, and *nnnn* is a unique number.

Note: A correlation ID passed to DB2 can be changed only by the CICS Attachment Facility issuing a signon to DB2. If signon reuse occurs by a thread using a primary authorization ID which remains constant across multiple transactions (for example, by using AUTHID(name)) only one signon will occur. In this instance the *tttt* in the correlation ID does not match the running transaction ID. It is the ID of the transaction for which the initial signon occurred.

If the thread is active within a unit of work, the CICS transaction name, its task number and finally the CICS local unit of work ID is displayed.

The correlation ID used in this display is also output on DB2 commands such as DISPLAY LOCK. For example, by using this display in conjunction with a display locks command you can find out which CICS task is holding a lock within DB2.

DISPLAY STATISTICS output

Output of a DSNC DISPLAY STATISTICS command is as follows:

```
DFHDB2014 07/09/98 14:35:45 IYK4Z2G1 Statistics report follows for RCTJT
accessing DB2 DB3A
                                                                -----COMMITS----
DB2ENTRY PLAN        CALLS    AUTHS    W/P HIGH   ABORTS   1-PHASE  2-PHASE
*COMMAND               1        1      1    1       0          0       0
*POOL    ********      4        1      0    1       0          2       0
XC05     TESTP05      22        1     11    2       0          7       5
XP05     ********      5        2      0    1       0          1       1
DFHDB2020 01/17/98 15:45:27 IYKA4Z2G1  The display command is complete.
```

Figure 7. Sample output from DSNC DISPLAY STATISTICS command

DB2ENTRY
Name of the DB2ENTRY, or '*COMMAND' for DSNC command calls, or '*POOL' for pool statistics.

PLAN
The plan name associated with this entry. Eight asterisks in this field indicate that this transaction is using dynamic plan allocation. The command processor transaction DSNC does not have a plan associated with it.

If a plan name associated with an entry is dynamically changed, the last plan name is the one put into use.

CALLS
The total number of SQL statements issued by transactions associated with this entry.

AUTHS
The total number of sign-on invocations for transactions associated with this entry. A sign-on does not indicate whether a new thread is created or an existing thread is reused. If the thread is reused, a sign-on occurs only if the authorization ID or transaction ID has changed.

W/P
The number of times that all available threads for this entry were busy. This value depends on the value of THREADWAIT for the entry. If THREADWAIT was set to POOL in the RCT, W/P indicates the number of times the transaction overflowed to the pool. An overflow to the pool shows up in the transaction statistics only and is not reflected in the pool statistics.

If THREADWAIT was set to YES, this reflects the number of times that the thread both had to wait, and could not attach a new subtask (the number of started tasks has reached THREADLIMIT).

The only time W/P is updated for the pool is when a transaction had to wait for a pool thread and a new subtask could not be attached for the pool. The W/P statistic is useful for determining if there are enough threads defined for the entry.

HIGH
The maximum number of threads acquired by transactions associated with this entry at any time since the connection was started, that is, a high watermark of threads.

Note: In releases before CICS Transaction Server 1.2, the HIGH value also included the transactions forced to wait for a thread or those diverted to the pool. Now the HIGH value *only* represents threads actually created on the entry.

ABORTS
The total number of units of recovery which were rolled back. It includes both abends and syncpoint rollbacks, including syncpoint rollbacks generated by -911 SQL codes.

COMMITS
One of the following two fields is incremented each time a DB2 transaction associated with this entry has a real or implied (such as EOT) syncpoint. Units of recovery that do not process SQL calls are not reflected here.

1-PHASE
The total number of single-phase commits for transactions associated with this entry. This total does not include any two-phase commits (see the explanation for 2-PHASE below). This total does include read-only commits as well as single- phase commits for units of recovery which have performed updates. A two-phase commit is needed only when the application has updated recoverable resources other than DB2.

2-PHASE
The total number of two-phase commits for transactions associated with this entry. This number does not include single phase commit transactions.

DSNC MODIFY

The CICS attachment facility command DSNC MODIFY modifies the message queue destinations of the DB2CONN, THREADLIMIT value for the pool, for DSNC commands, or for DB2ENTRYs. The same function can also be achieved using CEMT/EXEC CICS SET DB2CONN or DB2ENTRY commands.

Environment

This command can be issued only from a CICS terminal.

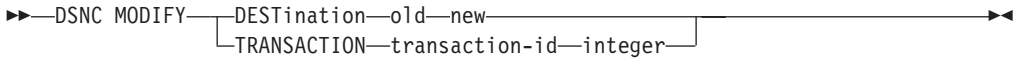
Abbreviation

DSNC MODI or MODI (using the MODI transaction from the CICS DB2 sample group DFH$DB2).

Authorization

Access to this command can be controlled using the following CICS authorization checks:

- Transaction attach security for transaction DSNC.
- Command security for resource DB2CONN. This command requires UPDATE access.
- For DSNC MODIFY TRANSACTION commands modifying the attributes of a DB2ENTRY. There are also command security checks for resources DB2ENTRY and DB2TRAN and resource security for the DB2ENTRY. The command requires READ access to resource DB2TRAN, and UPDATE access to resource DB2ENTRY for command security. In addition, the resource security command requires UPDATE access to the particular DB2ENTRY involved. For more information about CICS security, see “Chapter 2. Installation and migration considerations” on page 7.



Parameter description

DESTination
Specifies that the MSGQUEUE parameter of the DB2CONN table is to be changed, replacing the ″old″ destination ID with the ″new″ destination ID.

old Any destination ID currently set in the MSGQUEUE of the DB2CONN.

new A new destination identifier.

TRANsaction

Specifies that the THREADLIMIT value associated with the given transaction or group is to be modified.

transaction-ID
The command uses a transaction ID to identify either the pool, command, or the DB2ENTRY THREADLIMIT value to be modified.

- To change the THREADLIMIT value of the pool, a transaction ID of CEPL must be used.
- To change the THREADLIMIT for command threads, a transaction ID of DSNC must be used.
- To change the THREADLIMIT for a DB2ENTRY, use the transaction ID of any transaction that is defined to use the DB2ENTRY.

integer
Is a new maximum value.

Usage notes

The integer specified in the command DSNC MODIFY TRANSACTION cannot be larger than the value specified for the TCBLIMIT parameter of the DB2CONN. The lowest possible value is zero.

Examples

Example 1

To change the specification of the MSGQUEUE parameter in the DB2CONN from MTO1 to MTO2 as follows:

```
DSNC MODI DEST MTO1 MTO2
```

```
DFHDB2039 07/09/98 14:47:17 IYK4Z2G1 The error destinations are: MT02 ****
****.
```

Figure 8. Sample output from DSNC MODIFY DESTINATION command

Example 2

To change the pool thread limit to 12:

```
DSNC MODI TRAN CEPL 12
```

```
DFHDB2019 07/09/98 14:49:28 IYK4Z2G1 The modify command is complete.
```

Figure 9. Sample output from DSNC MODIFY TRANSACTION command (pool thread)

Example 3

To change the command thread limit to 3:

```
DSNC MODI TRAN DSNC 3
```

```
DFHDB2019 07/09/98 14:49:28 IYK4Z2G1 The modify command is complete.
```

Figure 10. Sample output from DSNC MODIFY TRANSACTION command (for a command thread)

Example 4

To change the thread limit of the DB2ENTRY used by transaction XP05 to 8:

```
DSNC MODI TRAN XP05 8
```

```
DFHDB2019 07/09/98 14:49:28 IYK4Z2G1 The modify command is complete.
```

Figure 11. Sample output from DSNC MODIFY TRANSACTION command (changing DB2ENTRY thread limit)

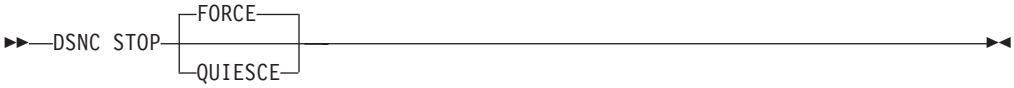
DSNC STOP

The CICS attachment facility command DSNC STOP stops the attachment facility. The same function can also be achieved by issuing a CEMT or EXEC CICS SET DB2CONN NOTCONNECTED command.

Environment

This command can be issued only from a CICS terminal.

Syntax



Abbreviation

DSNC STOP or STOP (using the STOP transaction from the CICS DB2 sample group DFH$DB2).

Authorization

Access to this command can be controlled using the following CICS authorization checks:

- Transaction attach security for transaction DSNC
- Command security for resource DB2CONN. This command requires UPDATE access. For more information about CICS security, see “Chapter 2. Installation and migration considerations” on page 7.

Parameter description

QUIESCE
Specifies that the CICS attachment facility is to be stopped after CICS transactions currently running complete. In releases before CICS Transaction Server for OS/390 Release 2, QUIESCE would only wait for active transactions to release their thread, which, typically, was at the end of a unit of work (UOW). Now QUIESCE waits for active transactions to complete, that is, new UOWs can start and acquire threads.

FORCE
Specifies that the CICS attachment facility is to be stopped immediately by forcing disconnection with DB2, regardless of any transactions that are running. Currently running transactions that have accessed DB2 are forcepurged. This includes transactions that may have committed updates to DB2 in a previous UOW, but have not yet accessed DB2 in their current UOW.

Usage notes

For a DSNC STOP QUIESCE, message DFHDB2012 is output to the terminal. The terminal then remains locked until shutdown is complete, when message DFHDB2025 is output.

For a DSNC STOP FORCE, message DFHDB2022 is output to the terminal. The terminal then remains locked until shutdown is complete, when message DFHDB2025 is output.

Examples

Example 1

To quiesce stop the CICS DB2 attachment facility:

```
DSNC STOP
```

```
DFHDB2012 07/09/98 14:54:28 IYK4Z2G1 Stop quiesce of the CICS-DB2 attachment
facility from DB2 subsystem DB3A is proceeding.
```

Figure 12. Sample output from DSNC STOP command

The message resulting from the DSNC STOP command shown in Figure 12 is replaced by the message shown in Figure 13 when shutdown is complete.

```
DFHDB2025I 07/09/98 14:58:53 IYK4Z2G1 The CICS-DB2 attachment has disconnected
from DB2 subsystem DB3A
```

Figure 13. Sample output from DSNC STOP when shutdown is complete

Example 2

To force stop the CICS DB2 attachment facility:

```
DSNC STOP FORCE
```

```
DFHDB2022 07/09/98 15:01:51 IYK4Z2G1 Stop force of the CICS-DB2 attachment
facility from DB3A is proceeding.
```

Figure 14. Sample output from DSNC STOP FORCE command

The message resulting from the DSNC STOP FORCE command shown in Figure 14 is replaced by the message shown in Figure 15 on page 33 when shutdown is complete.

```
DFHDB2025I 07/09/98 15:10:55 IYK4Z2G1 The CICS-DB2 attachment has disconnected
from DB2 subsystem DB3A
```

Figure 15. Sample output from DSNC STOP FORCE when shutdown is complete

DSNC STRT

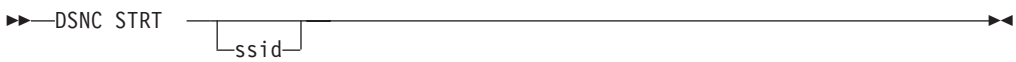
The DSNC STRT command starts the CICS attachment facility, which allows CICS application programs to access DB2 databases. The same function can also be achieved by issuing a CEMT or EXEC CICS SET DB2CONN CONNECTED command.

Environment

This command can be issued only from a CICS terminal.

Syntax

DSNC STRT syntax



Abbreviation

DSNC STRT or STRT (using the STRT transaction from the CICS DB2 sample group DFH$DB2).

Authorization

Access to this command can be controlled using the following CICS authorization checks:

- Transaction attach security for transaction DSNC
- Command security for resource DB2CONN. This command requires UPDATE access. For more information about CICS security, see “Chapter 2. Installation and migration considerations” on page 7.

Parameter description

ssid
Specifies a DB2 subsystem ID to override that specified in the DB2CONN.

Usage notes

If a DB2CONN is not installed when the DSNC STRT command is issued, error message DFHDB2031 is produced, indicating that no DB2CONN is installed. RCT resources must be installed from the CSD before attempting to start the CICS DB2 attachment facility.

APAR PQ36238

Documentation change. APAR PQ 36238 MJO 02/08/00

The hierarchy for determining the DB2 subsystem to use is as follows:

1. Use the subsystem ID if specified in a DSNC STRT command
2. Use the DB2ID in the installed DB2CONN if not blank
3. Use the subsystem ID if specified on INITPARM when the DB2ID in the last installed DB2CONN is blank (or has subsequently been set to blanks). On any startup, INITPARM is always used if the last installed DB2CONN contained a blank DB2ID even if the DB2ID has been consequently changed using a SET command.
4. Use a default subsystem ID of DSN.

Examples

Example 1

To start the CICS DB2 attachment facility using an installed DB2CONN:

```
DSNC STRT
```

```
DFHDB2023I 07/09/98 15:06:07 IYK4Z2G1 The CICS DB2 attachment has connected to
DB2 subsystem DB3A
```

Figure 16. Sample output from DSNC STRT command

Example 2

To start the CICS DB2 attachment facility using an installed DB2CONN but overriding the DB2 subsystem ID with DB3A:

```
DSNC STRT DB3A
```

```
DFHDB2023I 07/09/97 15:06:07 IYK4Z2G1 The CICS DB2 attachment has connected to
DB2 subsystem DB3A
```

Figure 17. Sample output from DSNC STRT using DB3A

If DB2 is not active when an attempt is made to start the CICS DB2 attachment facility, the following output is received if STANDBYMODE=NOCONNECT is specified in the DB2CONN:

```
DFHDB2018 07/09/98 15:14:10 IYK4Z2G1 DB3A DB2 subsystem is not active.
```

Figure 18. Sample output from DSNC STRT when DB2 is not active

If STANDBYMODE=CONNECT or RECONNECT, the following output is received:

```
DFHDB2037 07/09/98 15:15:42 IYK4Z2G1 DB2 subsystem DB2A is not active.
CICS DB2 attachment facility is waiting.
```

Figure 19. Alternative output from DSNC STRT when DB2 is not active

Chapter 5. Defining the CICS DB2 connection

This chapter discusses the connection between CICS and DB2 in the following sections:

CICS Transaction Server Release 2 was the last CICS release to support operating the CICS DB2 interface with an RCT. In releases of CICS before CICS Transaction Server Release 2, the connection between CICS and DB2 was defined in the resource control table (RCT).

Resource definition online (RDO) now defines DB2CONN, DB2ENTRY and DB2TRAN objects. These objects describe the relationship between CICS transactions and DB2 resources (application plans and command processors), and in DB2CONN, also describe the global attributes of the CICS DB2 connection.

For detailed information about the description of the DB2CONN, DB2ENTRY and DB2TRAN objects, and the parameters they take, see the *CICS Resource Definition Guide*

When describing parameters of the CICS DB2 connection the names of the parameters of the RDO objects are used, not the DSNCRCT macro parameter names. The *CICS Resource Definition Guide* describes which DSNCRCT macro parameter applies to which RDO parameter, and which RDO parameter applies to which DSNCRCT macro parameter.

CICS DB2 thread types

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

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 are transferred to the pool if a DB2 command thread is not available.

Command threads are defined in the command threads section of the DB2CONN.

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 "Coordinating DB2CONN, DB2ENTRY, and BIND options" on page 41 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 is waiting to use them. The overhead of creating and terminating the thread must be taken into account for transactions using unprotected entry threads.

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

Entry threads are defined using a DB2ENTRY definition.

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 if no CICS transaction is waiting to use it.

Pool threads are defined in the pool threads section of the DB2CONN.

MVS subtasks

For the three types of thread, the thread runs under an MVS subtask. These thread subtasks are 'daughter' subtasks of the main CICS DB2 subtask DFHD2MSB established when CICS connects to DB2. DFHD2MSB is a subtask of the main CICS TCB, and hence the thread subtasks are 'grand daughters' of the main CICS TCB.

The number of thread subtasks allowed is controlled using the TCBLIMIT parameter of the DB2CONN. An MVS subtask is not terminated when the thread is terminated. A thread subtask can be terminated if:

- A CICS transaction is force purged from CICS and the thread is still active in DB2. In this case the subtask is terminated as a means of flushing the request out of DB2. The current UOW in DB2 is backed out.
- The TCBLIMIT value of the DB2CONN is lowered. When a thread is terminated, the CICS DB2 attachment recognises that the current number of TCBs exceeds the TCBLIMIT, and terminates the TCB as well.
- CICS is indoubt as to the outcome of a UOW because it has lost contact with its coordinator. Terminating the subtask causes DB2 to release the thread, but maintain the UOW as indoubt and maintain its locks. The UOW is completed by a later resynchronization when CICS reestablishes contact with its coordinator.
- The CICS DB2 attachment facility is shut down.

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:

- DB2CONN and DB2ENTRY specifications:
  - Use of protected entry threads
  - Use of unprotected entry or pool threads
  - Authorization strategy
- BIND options
  - ACQUIRE
  - RELEASE
  - VALIDATE
- Use of PACKAGES

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 database 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 DB2ENTRY 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. For more information, see “Locking” on page 43.

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 an 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.

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 the 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).

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 SYNCPOINT statement.

Transactions release the thread they are using at different times. If the transaction is terminal-oriented, or non-terminal-oriented and NONTERMREL=YES is specified in the DB2CONN, the thread is released at SYNCPOINT as well as at end of task (EOT). This makes it efficient to use a protected thread for transactions issuing many SYNCPOINTS, 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 syncpoint:

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

Threads are not released at SYNCPOINT time if:

- Held cursors are open.

- The following DB2 modifiable special registers are not at their initial value:
  CURRENT PACKAGESET
  CURRENT SQLID
  CURRENT SERVER

  DB2 unmodifiable special registers are:
  CURRENT DATE
  CURRENT TIME
  CURRENT TIMESTAMP
  CURRENT TIMEZONE
  CURRENT USER
- The DB2 modifiable special register, CURRENT DEGREE, is ever changed during the lifetime of the CICS task.

If the transaction is not terminal-oriented and you specify NONTERMREL=NO, the thread is released at EOT only. 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 syncpoints.

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.

Coordinating DB2CONN, DB2ENTRY, and BIND options

You can create many different combinations of DB2CONN, DB2ENTRY, and 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 DB2CONN, DB2ENTRY, and BIND options.

Recommended combinations

In general it is recommended that you initially set these options to the values shown in Table 1. You may find that you get better performance from other combinations for your own transactions. For each transaction type a recommended thread type and BIND option are shown. There are also recommendations whether transactions should overflow to the pool.

Table 1. CICS DB2 transaction types

Transaction Description	Thread Type	Overflow	ACQUIRE	RELEASE
High volume (all types)	Protected Entry	Note 1	ALLOCATE	DEALLOCATE
Terminal-oriented with many commits (plus non-terminal if NONTERMREL=YES)	Protected Entry	Note 2	Note 3	DEALLOCATE
Low volume, high priority	Unprotected Entry	Yes	USE	COMMIT

Table 1. CICS DB2 transaction types (continued)

Transaction Description	Thread Type	Overflow	ACQUIRE	RELEASE
Low volume, limited concurrency	Unprotected Entry	Never	USE	COMMIT
Low volume, low priority	Pool	Not Applicable	USE	COMMIT
Non-terminal-oriented with many commits (NONTERMREL=NO)	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 1 on page 41 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 1 on page 41 shows, only a few combinations of DB2CONN, DB2ENTRY, and BIND options are generally recommended. However, in specific situations other combinations can be used.

Processing SQL requests

Table 2 shows a summary of the activities involved in processing SQL requests for the three recommended sets of DB2CONN, DB2ENTRY, and BIND specifications.

Table 2. Thread-Related Activities

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:				

Table 2. Thread-Related Activities (continued)

Activity	Protected Threads		Unprotected Threads	
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
2. Only if SQL section is not already in EDM pool
3. Only if SQL section is not already in Cursor Table

Table 2 on page 42 illustrates the performance advantage of using protected threads without changing the authorization ID.

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 is not 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.

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 a DB2ENTRY entry with PROTECTNUM>0 should use the same authorization ID. At least they should avoid specifying TERM, OPID, and USERID for the AUTHTYPE parameter, because these values often vary among instances of a transaction.

Grouping transactions

Several transactions can be specified to use the same DB2ENTRY. Ideally, they should all use the same plan. Each low-volume transaction can have its own DB2ENTRY. In this case thread reuse is not likely and you should specify PROTECTNUM=0. An alternative is to group a number of low-volume transactions in the same DB2ENTRY and specify PROTECTNUM=n. This gives better thread utilization and less overhead.

Creating, using, and terminating threads

This section deals with the processes involved in creating, using, and terminating threads, and attaching thread TCBs, and the implications of these on specifying DB2CONN and DB2ENTRY parameters. Overflow to the pool is also discussed. The section looks 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 using DB2CONN for pool threads, and DB2ENTRY for entry threads.

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

- When the CICS attachment facility is started no thread TCBs are started. They are started as needed when work arrives.
- 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.
- When a thread is created and another transaction with a new authorization ID is reusing a thread, DB2 makes an authorization check for 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, unless NONTERMREL=YES is specified in the DB2CONN.
- When a transaction releases a thread, the thread can be reused by another transaction specifying the same plan and defined in the same DB2ENTRY. 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.
- A protected thread is terminated if not used for two consecutive periods of 30 seconds. This averages about 45 seconds. This value can be modified by the PURGECYCLE parameter of the DB2CONN.
- The THREADWAIT parameter defines whether the requests for a thread should be queued, abended, or overflowed to the pool in case of entry thread shortage.

Protected entry threads

These threads are defined with the following DB2ENTRY parameters:

```
PROTECTNUM(n)
THREADLIMIT(n)
```

Protected entry threads are recommended for:

- High-volume transactions of any type
- Terminal-oriented transactions with many commits
- Non-terminal-oriented transactions with many commits (if NONTERMREL=YES is specified in the DB2CONN)

Protected entry threads are created, used, and terminated as follows:

TCB attach
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 if an existing thread is not available.

If no thread is available, but an unused TCB exists for this DB2ENTRY, a new thread is created and related to the TCB, provided THREADLIMIT is not exceeded.

If no thread is available and no unused TCB is available, a new TCB is attached, and a new thread is built provided the TCBLIMIT and THREADLIMIT are not exceeded.

Thread termination
If the current number of protected threads is less than the PROTECTNUM value when the thread is released and there is no new work queued, the thread is marked as protected. Otherwise it is terminated. A protected thread is terminated if it is unused for two consecutive 30 second periods or PURGECYCLE value.

Thread reuse
Other transactions using the same DB2ENTRY may reuse a thread, if it is available. This is likely because the threads remain active for about 45 seconds after being released, depending on the PURGECYCLE value.

Overflow to pool
If THREADWAIT=POOL is specified, requests for threads are transferred to the pool when the value of THREADLIMIT is exceeded. A transaction is then under the control of the PRIORITY, THREADLIMIT, and THREADWAIT parameters for the pool. The transaction keeps the PLAN and the AUTHID/AUTHTYPE values you specified for the entry thread.

High priority unprotected entry threads

These threads are defined with the following DB2ENTRY parameters:

```
PROTECTNUM(0)
THREADLIMIT(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 DB2ENTRY, a new thread is created and related to the TCB, provided THREADLIMIT is not exceeded.

If the limit for THREADLIMIT and TCBLIMIT 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 to use the same DB2ENTRY can reuse a thread, if it is available. This happens only if a transaction is waiting for the thread when the thread becomes available.

Overflow to pool
If THREADWAIT=POOL is specified, requests for threads are transferred to the pool when the value of THREADLIMIT is exceeded. A transaction is then under the control of the PRIORITY, THREADLIMIT, and THREADWAIT parameters for the pool. The transaction keeps the PLAN and the AUTHID/AUTHTYPE values you specified for the entry thread.

Note that you should not allow overflow to the pool for limited concurrency transactions.

Low priority unprotected entry threads

These threads are defined with the following DB2ENTRY parameters:

```
PROTECTNUM(0)
THREADLIMIT(0)
THREADWAIT(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 THREADLIMIT=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 PRIORITY, THREADLIMIT, and THREADWAIT parameters for the pool. The transaction keeps the PLAN and the AUTHID/AUTHTYPE values you specified for the entry thread.

Thread creation
A thread is created in the pool when needed by a transaction unless the THREADLIMIT 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.

Pool threads

These threads are defined with the following DB2CONN parameter:

```
THREADLIMIT(n)
```

There are four cases when a pool thread can be used:

1. A transaction is not specified in any DB2ENTRY or DB2TRAN, but issues SQL requests. This transaction defaults to the pool and uses the plan specified for the pool.
2. A transaction is specified in a DB2ENTRY or DB2TRAN referencing a DB2ENTRY, but is forced to the pool because the DB2ENTRY specifies THREADLIMIT(0) and THREADWAIT(POOL). This transaction uses the plan specified in the DB2ENTRY.
3. A transaction is specified in a DB2ENTRY or DB2TRAN referencing a DB2ENTRY, but overflows to the pool (THREADWAIT=POOL) when the THREADLIMIT value is exceeded. This transaction uses the plan specified in the DB2ENTRY.
4. A transaction is specified in a DB2ENTRY or a DB2TRAN referencing a DB2ENTRY, but the DB2ENTRY is disabled. The DISABLEDACT keyword is set to POOL, therefore a pool thread is used. This transaction uses the plan specified in the DB2ENTRY.

Pool threads are always unprotected threads.

Pool 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 when 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.

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.

Chapter 6. Security

Security for CICS DB2 is discussed in this chapter in the following sections:

Note: In this chapter, we refer to RACF® as the external security manager (ESM) used by CICS. Except for the explicit RACF examples, the general discussion applies equally to any functionally equivalent non-IBM ESM.

Overview

Several different security mechanisms can be used to control access to resources in a CICS DB2 environment:

Dataset protection
You can use 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.

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

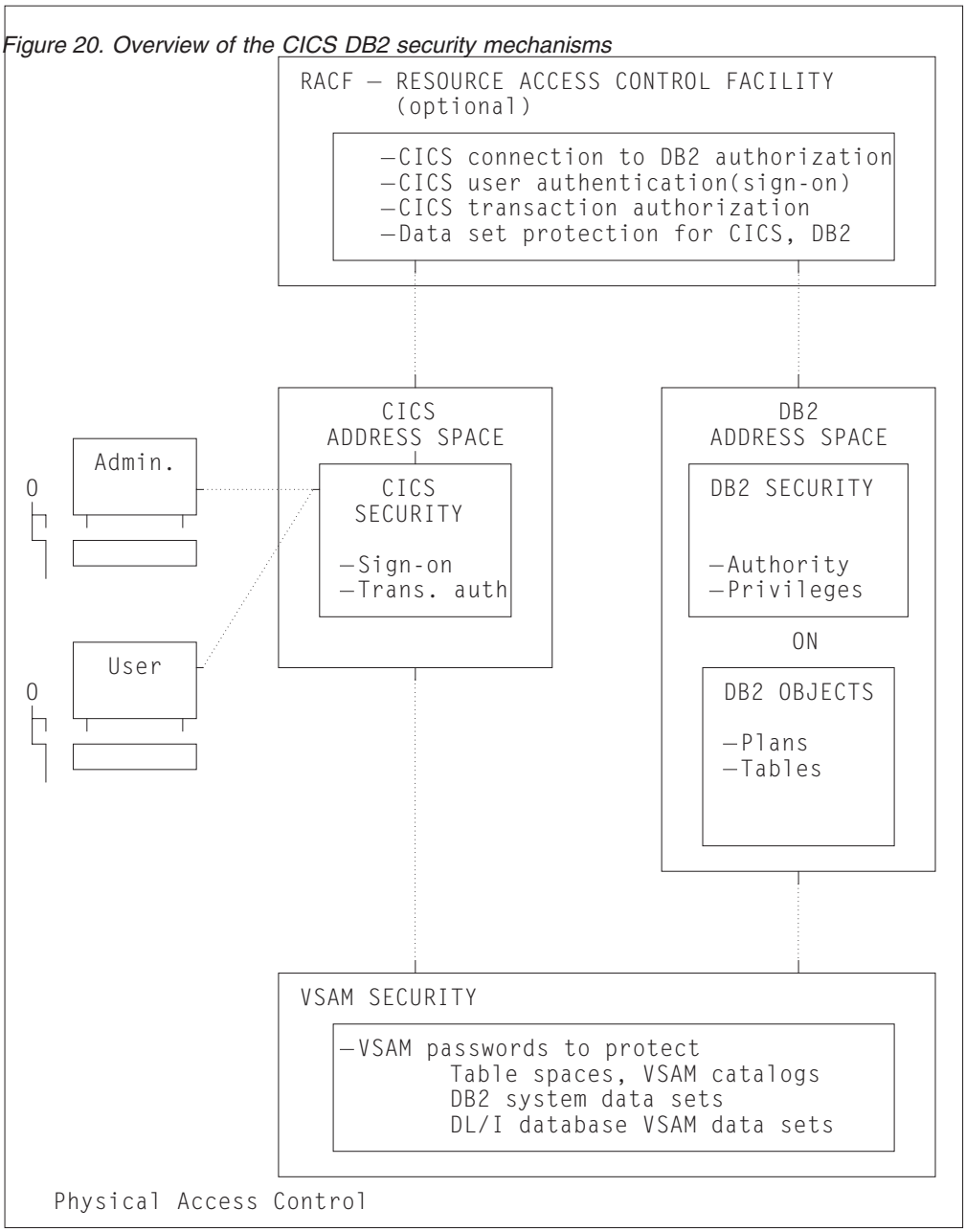
User authentication
CICS sign-on authenticates users by checking that they supply a valid user ID and password. Note that DB2 does not authenticate the user; this function is performed by CICS.

CICS security
Checks that the authenticated user ID is authorized to use a specific transaction. This function is also sometimes referred to as *transaction-attach security*, which is described in *CICS RACF Security Guide*. CICS security extends to checking that the user ID is authorized to issue CICS SPI commands that operate on DB2 resource definitions. See "CICS security" on page 52 for information about the various security checks that CICS performs.

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

Figure 20 shows the security mechanisms involved in a CICS DB2 environment.

Figure 20. Overview of the CICS DB2 security mechanisms



RACF 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 region's user ID and the requested connection type. For CICS the connection type is single address space subsystem (SASS). RACF checks that the CICS region user ID, which is the initial primary authorization ID, is authorized to connect to DB2 (see

"Primary and secondary authorization IDs" on page 58 for information about primary authorization IDs). The region user ID under which CICS executes is one of the following:

- The user ID taken from the RACF started procedures table, ICHRIN03, if CICS is running as a started task.
- The user parameter of the STDATA segment in a STARTED general resource class profile, if CICS is running as a started job.
- The user ID specified on the USER parameter of the JOB card, if CICS is running as a job.

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 subsystem DB2A in class DSNR.

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

2. Permit CICS to access the specified DB2 subsystem.

   For example, the following RACF command permits CICS system CICSHA11 to connect to DB2 sub system DB2A:

   ```
   PERMIT DB2A.SASS CLASS(DSNR) ID(CICSHA11) 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 (see "Primary and secondary authorization IDs" on page 58 for information about secondary authorization IDs). 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, 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.

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

- You did not specify AUTHTYPE(GROUP) in the DB2ENTRY or DB2CONN
- 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 is not defined to RACF as a user profile.

In summary, a CICS region, using a specification other than AUTHTYPE(USERID) or AUTHTYPE(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 the *DB2 for OS/390: Administration Guide*.

User authentication

User authentication is achieved by signing on to CICS to authenticate your user ID. If you do not sign on, CICS uses the default user ID unless a user ID has been permanently associated with a terminal using preset security. The authenticated user ID is used when any of the following AUTHTYPEs are specified in a DB2ENTRY or the DB2CONN:

USERID
OPID
GROUP

CICS security

CICS transactions that issue DB2 commands or SQL commands can involve security checks in two address spaces—the CICS region and the DB2 sub system. The security checks performed by CICS are as follows:

Transaction attach security
CICS calls RACF at transaction attach time to check that the user ID associated with the transaction is authorized to run the transaction.

CICS command security
If the transaction contains SPI commands that operate on DB2CONN, DB2ENTRY, or DB2TRAN resource definitions, CICS calls RACF to check that the user ID is authorized to issue SPI commands against these resource types.

CICS resource security
If the transaction contains commands that reference a DB2ENTRY, or a DB2TRAN, CICS calls RACF to check that the user ID is authorized to use that DB2ENTRY.

Surrogate security
If the transaction contains SPI commands that reference the DB2 authorization IDs, AUTHID or COMMAUTHID, CICS calls RACF to check that the user ID associated with the transaction is authorized as a surrogate of the DB2 authorization ID.

AUTHTYPE security
If the transaction contains SPI commands that reference the DB2 authorization IDs AUTHTYPE or COMMAUTHTYPE, CICS calls RACF to check that the user ID associated with the transaction is authorized through an appropriate FACILITY class general resource profile.

For more information about the CICS security topics discussed here, such as command and resource security in general, see the *CICS RACF Security Guide*.

Transaction attach security

CICS checks that your user ID is allowed to use a specific transaction. Ensure that DSNC and any other transactions that you defined for specific CICS attachment facility tasks or DB2 commands are protected. You should also protect the transactions that invoke your own CICS DB2 application programs.

CICS command security

You activate CICS command security globally in a CICS region by specifying XCMD=YES (or XCMD=*profile_class_name*) as a system initialization parameter. You control command security at the individual transaction level by specifying CMDSEC(YES) on the transaction resource definition.

If command security is active in a region, and specified on a transaction that issues EXEC CICS INQUIRE, SET, CREATE or DISCARD commands against DB2 resource definitions, users must be authorized to access the DB2 resource names.

To protect the DB2 resource definitions using command security, add the DB2 resource names DB2CONN, DB2ENTRY, and DB2TRAN as resource identifiers in one of the RACF default resource class profiles for CICS commands, CCICSCMD or VCICSCMD. Alternatively, you can use a user-defined general resource class that is defined on the XCMD system initialization parameter. See the *CICS System Definition Guide* for information about specifying CICS command profile class names on the XCMD parameter.

Permit access to these DB2 resource names for authorized users by giving the appropriate authority as follows:

INQUIRE command
: Requires READ authority

SET command
: Requires UPDATE authority

CREATE command
: Requires ALTER authority

DISCARD command
: Requires ALTER authority

In the following example, the RDEFINE command defines a profile named CMDSAMP in the default class VCICSCMD. The DB2 resource types protected by this profile are specified on the ADDMEM operand. The PERMIT command allows a group of users to issue EXEC CICS INQUIRE commands against the specified DB2 object types.

```
RDEFINE  VCICSCMD CMDSAMP UACC(NONE)
                  NOTIFY(sys_admin_userid)
                  ADDMEM(DB2CONN, DB2ENTRY, DB2TRAN)
PERMIT CMDSAMP CLASS(VCICSCMD) ID(operator_group) ACCESS(READ)
```

Within a transaction, you can query whether a user ID has access to these DB2 resource types by using the EXEC CICS QUERY RESTYPE(SPCOMMAND) command, with the RESID parameter specifying DB2CONN, DB2ENTRY, or DB2TRAN.

CICS resource security

CICS resource security checking verifies that a user is authorized to access the specific resource referenced by an EXEC CICS command. CICS also uses resource security to control access to specific DB2ENTRY definitions referenced by EXEC CICS INQUIRE, SET, DISCARD, and CREATE commands. Protecting DB2ENTRYs also protects associated DB2TRAN resources definitions, which are regarded as an extension of a DB2ENTRY.

You activate CICS resource security globally in a CICS region by specifying the system initialization parameter for a specific resource: for DB2, you specify XDB2=*profile_class_name*. You control resource security at the individual transaction level by specifying RESSEC(YES) on the transaction resource definition.

CICS performs resource security checking against SPI commands that reference specific DB2ENTRYs. This means that, if you specify CMDSEC=YES and

RESSEC=YES on transactions that issue SPI commands against DB2ENTRYs, CICS performs two security checks against the user ID:

1. That the user ID is authorized to issue SPI commands against DB2CONN, DB2ENTRY, and DB2TRAN resource types.
2. That the user ID is authorized to access a specific DB2ENTRY.

You permit access to DB2ENTRYs by giving users the appropriate access, as follows:

INQUIRE command
: Requires READ authority

SET command
: Requires UPDATE authority

CREATE command
: Requires ALTER authority

DISCARD command
: Requires ALTER authority

Defining RACF general resource class profiles for DB2

You protect DB2ENTRYs by defining profiles for the DB2ENTRYs in a suitable general resource class.

Unlike the RACF default resource class names for CICS, there are no IBM-supplied default class names for DB2ENTRYs. Create your own installation-defined classes for DB2ENTRYs by adding new class descriptors to the installation-defined part (module ICHRRCDE) of the RACF class descriptor table (CDT). For an example of how to do this, see the IBM-supplied sample job, RRCDTE, supplied in member DFH$RACF of CICSTS13.CICS.SDFHSAMP. This gives an example of a member class called XCICSDB2, and a grouping class called ZCICSDB2. This example uses the same naming convention as the default resource class names for CICS. You are recommended to create new class names using a similar naming convention rather than use existing CICS class names for DB2ENTRYs.

In the following example, the RDEFINE command shows how to add a number of DB2ENTRY names to the XCICSDB2 resource class:

```
RDEFINE  XCICSDB2  (db2ent1, db2ent2, db2ent3.., db2entn) UACC(NONE)
                   NOTIFY(sys_admin_userid)
```

Note: Protecting DB2ENTRY resource definitions also protects access to associated DB2TRAN definitions, a DB2TRAN being considered an extension to the DB2ENTRY to which it refers.

Authorizing access to a DB2ENTRY

When you have created the required profiles in the XCICSDB2 resource class, the next step is to authorize users, or groups of users, to inquire on, modify, discard, or create protected DB2ENTRY resource definitions.

The following example shows how to authorize a group of users to modify a protected DB2ENTRY using an EXEC CICS SET command:

```
PERMIT db2ent1 CLASS(XCICSDB2) ID(group1) ACCESS(UPDATE)
```

The following example shows how to authorize a group of users to discard a protected DB2ENTRY using an EXEC CICS DISCARD command:

```
PERMIT db2ent2 CLASS(XCICSDB2) ID(group2) ACCESS(ALTER)
```

DB2TRAN resource security

If a transaction subject to resource security issues an EXEC CICS SET DB2TRAN command, CICS performs a security check to verify that the user ID associated with the transaction has the necessary authority for the DB2ENTRY to which the DB2TRAN refers. Thus, for resource security checking purposes, a DB2TRAN is an extension of the associated DB2ENTRY.

One effect of this is a double security check in the case where a transaction issues an EXEC CICS SET DB2TRAN(*name*) DB2ENTRY(*name*) to change the name of the associated DB2ENTRY. CICS issues two calls to RACF:

1. To verify that the user ID has the necessary authority for the old DB2ENTRY referenced by the DB2TRAN being modified.
2. To verify that the user has the necessary authority for the new DB2ENTRY specified on the SET DB2TRAN command.

In this case, the command is in effect modifying the attributes of two DB2ENTRYs.

Surrogate security

CICS performs surrogate user security checking using the surrogate user facility of RACF. A surrogate user is one who has the authority to do work on behalf of another user. A surrogate user is authorized to act for that user without knowing that other user's password.

CICS protects DB2 authorization IDs by checking that the user ID performing the
change is authorized as a surrogate of the authorization ID specified on an AUTHID
or COMAUTHID parameter. This also applies to the CICS region userid when such
resources are installed as part of a cold or initial start. In the case of an EXEC
CICS SET command that changes an authorization ID, the user ID making the
change must be a surrogate of the new authorization ID. This requires that
surrogate user checking is active in the CICS region.

You control surrogate user checking globally in a CICS region by specifying XUSER=YES as a system initialization parameter. If surrogate user checking is in force, CICS calls RACF to verify that user IDs are authorized as surrogates for the DB2 authorization IDs specified on AUTHID or COMAUTHID parameters.

The commands subject to surrogate user checks are EXEC CICS SET, CREATE, and INSTALL that operate on DB2CONN and DB2ENTRY resource definitions.

To define a user ID as a surrogate of a DB2 authorization ID, give the user ID READ authority to the authorization ID's profile in the RACF SURROGAT class. Create a profile for the authorization ID with a name of the form *authid*.DFHINSTL, with the authorization ID defined as the owner. In the following example, DB2AUTH1 is the authorization ID specified on the AUTHID or COMAUTHID parameter of a DB2CONN resource definition, or the AUTHID parameter specified on a DB2ENTRY resource definition:

```
RDEFINE  SURROGAT  DB2AUTH1.DFHINSTL  UACC(NONE) OWNER(DB2AUTH1)
```

Use the following command to permit DB2USER9 to act as a surrogate for DB2AUTH1:

```
PERMIT   DB2AUTH1.DFHINSTL  CLASS(SURROGAT) ID(DB2USER9)  ACCESS(READ)
```

AUTHTYPE security

CICS also uses RACF to control access to DB2 authorization IDs specified on the AUTHTYPE and COMAUTHTYPE parameters specified on a DB2CONN and AUTHTYPE specified on a DB2ENTRY. Although this form of security check is not a surrogate security check as in the case of AUTHID and COMAUTHID, it is globally controlled in the CICS region through the XUSER system initialization parameter.

Any user ID using SET, CREATE, or INSTALL commands that operate on a DB2CONN, DB2ENTRY, or DB2TRAN could be subject to an 'authype' security check. In all cases, user IDs require READ authority to an appropriate profile in the RACF FACILITY general resource class, where the profile names are of the form DFHDB2.AUTHTYPE.*authname*, where *authname* is the name of the DB2CONN or DB2ENTRY resource definition owning the profile.

The following example shows how to define the profile and give the required READ authority to enable user ID DB2USER2 to install or modify AUTHTYPE or COMAUTHTYPE on a DB2CONN resource definition named DB2CONN1:

```
RDEFINE  FACILITY  DFHDB2.AUTHTYPE.DB2CONN1   UACC(NONE)
PERMIT  DFHDB2.AUTHTYPE.DB2CONN1 CLASS(FACILITY) ID(DB2USER2) ACCESS(READ)
```

In the following example, authority is given to user ID DB2USER3 to modify or install the AUTHTYPE on a DB2ENTRY named DB2ENT5:

```
RDEFINE  FACILITY  DFHDB2.AUTHTYPE.DB2ENT5   UACC(NONE)
PERMIT  DFHDB2.AUTHTYPE.DB2ENT5 CLASS(FACILITY) ID(DB2USER3) ACCESS(READ)
```

CICS DB2 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.) Thus, any user who is authorized by CICS transaction security to use the DSNC transaction can issue these commands.

DFHD2CM1, which handles both CICS attachment facility and DB2 commands, can be activated by transactions other than DSNC. Thus you can give a different transaction code to each CICS attachment facility command.

Alternative transaction definitions using the following names are supplied in sample group DFH$DB2:

- DISC
- DISP
- STRT
- STOP
- MODI

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 also to DB2 security checking. If you are authorized by

CICS transaction security to use the DSNC transaction you can issue these DB2 commands, but DB2 then checks that you are authorized for the commands (see "DB2 security").

DFHD2CM1, which handles both CICS attachment facility commands and DB2 commands, can be activated by transactions other than DSNC. Thus you can give a different transaction code to each DB2 command:

-ALT	-CAN	-ARC
-DIS	-MOD	-REC
-RES	-SET	-STA
-STO	-TER	

Alternative transaction definitions for these transaction identifiers are supplied in sample group DFH$DB2.

You can give different CICS users authorization to subsets 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 of the DB2 commands.

You can distinguish DB2 commands from CICS attachment facility commands by the hyphen (**-**) 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 use the DSNC transaction (or one of the other transactions that invokes DFHD2CM1). DB2 security then checks that you are authorized to enter the particular DB2 command specified as data in the DSNC transaction.

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.

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 APPLID). The authorization ID passed to DB2 is determined by the AUTHID or AUTHTYPE parameters of the DB2ENTRY and DB2CONN. AUTHID specifies an 8-character string. Valid values for AUTHTYPE are:

USERID
Your CICS RACF user ID used in CICS signon

OPID Your CICS operator ID, defined in the CICS segment of your RACF user profile

TERM Your terminal ID

TX The transaction ID

SIGN The SIGNID from DB2CONN

GROUP
The RACF group ID for the signed-on user ID

Specifying AUTHTYPE(GROUP) is similar to the use of AUTHTYPE(USERID), but passes additional information to the DB2 sign-on authorization exit routine. AUTHTYPE(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 DB2 master subtask 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.

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, users 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 operand of the AUTHTYPE parameter in the DB2ENTRY or DB2CONN. 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 parameter
- Pass the RACF userid and the RACF group ID character strings to DB2 to be used for privilege and authority checking.

Note: You must replace the default DB2 authorization exit routines if you want to use secondary authorization IDs.

The default authorization exit routines pass only a primary authorization ID to DB2.

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 the DB2 SDSNSAMP library.

If you specify AUTHTYPE(GROUP) in a DB2ENTRY or DB2CONN definition, 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.

See the *DB2 for OS/390: Administration Guide* for more information about:

- Privileges and authorities, with regard to controlling access to DB2 objects
- Connection and sign-on exits and writing exit routines
- Connection and the sign-on process, with regard to processing connections and processing sign-ons

Establishing primary and secondary authorization IDs

To use the GROUP option (and thus establish a secondary authorization ID):

- Specify SEC=YES as a system initialization parameter to ensure CICS uses RACF (or an equivalent ESM). Also specify SECPRFX=YES if your CICS transaction profile names are defined using the CICS region user ID as a prefix.
- If the CICS region is using MRO, ensure each connected CICS region is also using RACF security.
- If you are using MRO, specify ATTACHSEC=IDENTIFY on the CONNECTION definition for the TOR to ensure that sign-on information is propagated from the TOR to the AORs.

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 is not specified, the secondary authorization ID is set to blanks.

Table 3 on page 60 shows the primary and secondary authorization IDs that the CICS attachment facility passes to DB2. CICS has already checked that the terminal user is authorized to run the transaction by this stage.

Table 3. Effect of the AUTHTYPE and AUTHID parameters

AUTHTYPE	Primary authorization ID	Secondary authorization ID
GROUP	CICS user ID	RACF groups
USERID	CICS user ID	blanks
OPID	CICS operator ID	blanks
SIGN	DB2CONN SIGNID parm	blanks
TERM	CICS terminal ID	blanks
TX	CICS transaction ID	blanks
AUTHID=character_string	character_string	blanks

Note: `character_string` can be up to eight characters in length from the allowed character set comprising A-Z 0-9 ¢ @ and # (lowercase letters a to z are converted to uppercase). For example, you could define `AUTHID=tester2`. In this case, the CICS attachment facility passes the characters `TESTER2` to DB2 as the primary authorization ID. AUTHID and AUTHTYPE are mutually exclusive.

Authorizing commands to DB2

"DB2 command authorization" on page 56 describes how CICS checks whether you are authorized to use a CICS transaction that invokes 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 specify COMAUTHTYPE(USERID) on the DB2CONN definition, which resolves to the 8-byte CICS user ID as the authorization ID in DB2. Using this method, you can give different DB2 privileges explicitly to CICS user IDs. For example, you can use GRANT DISPLAY to give specific CICS user IDs permission to use only the -DIS command.

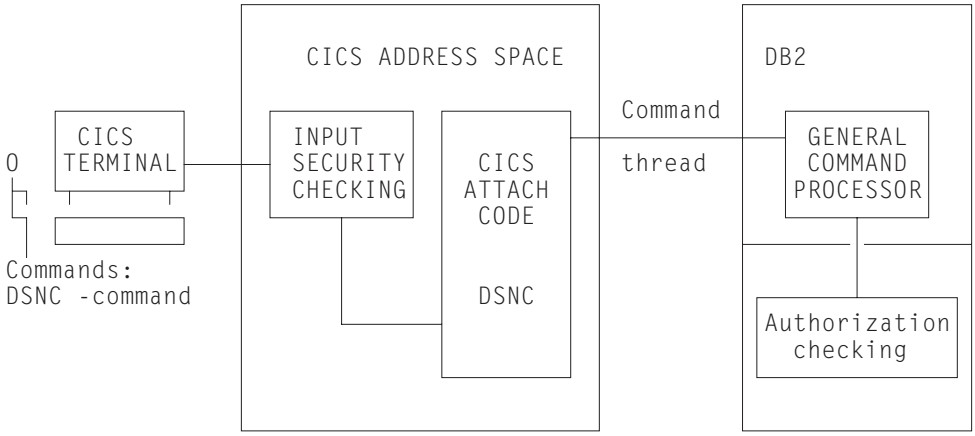


Figure 21. Security mechanisms for DB2 commands

Figure 21 shows the security mechanisms for DB2 command authorization.

- 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 SEC, SECPRFX, and XTRAN system initialization parameters to control CICS transaction attach security. For more information, see the *CICS RACF Security Guide*.
- The COMAUTHID or COMAUTHTYPE parameter on the DB2CONN.
- The DB2 system operator (SYSOPR) authority, which a user must have in order to use DB2 commands.

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 signed-on user ID is authorized to run the CICS-DB2 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 DB2ENTRY or DB2CONN. It can be any one of the authorization IDs listed in section “DB2 authorization IDs” on page 57. Figure 22 on page 62 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.

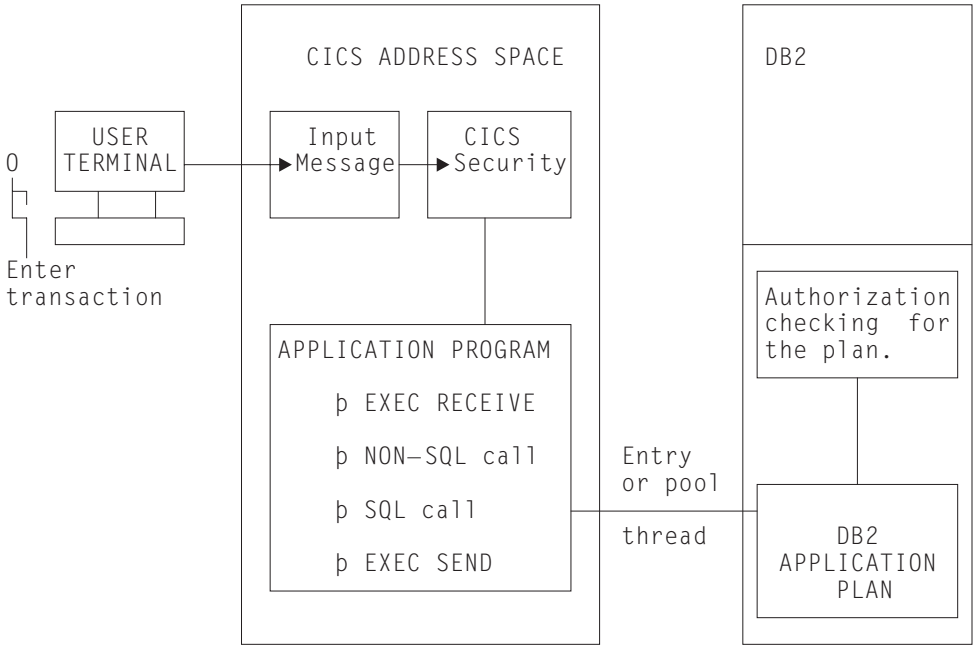


Figure 22. Security mechanisms for executing a plan (static SQL)

Performance recommendation

If you have a large terminal network with many users, it is recommended that you avoid using the following authorization IDs:

USERID
OPID
TERM
GROUP

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

- 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 AUTHTYPE of SIGN, TX, or AUTHID(*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 special register, CURRENT DEGREE, has ever been changed, or if there is an open cursor with hold. If you specify ACCOUNTREC(UOW) in the DB2ENTRY or DB2CONN, 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.

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 a sign-on occurs.

You can use the CEMT or EXEC CICS SET DB2CONN SECURITY(REBUILD) command to cause the CICS DB2 attachment facility to issue a DB2 sign-on the next time a thread is reused, or when a thread is built on an already signed-on TCB. This ensures that DB2 is made aware of the security change.

Accounting

The authorization ID is used in each DB2 accounting record. The value of the authorization ID is the chosen identifier in the AUTHID or AUTHTYPE parameter. Depending on the ACCOUNTREC 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, OPID, 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 9. Accounting” on page 79.

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 AUTHID or AUTHTYPE parameter of the DB2ENTRY or DB2CONN. The synonym must be created under the authorization ID of the original binder.
5. During the bind process at execution time, DB2:
   - Checks authorization
   - Selects path

- Generates execution code

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

It is recommended that you do not use both synonyms and VALIDATE(RUN) together.

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 AUTHTYPE(*userid*), 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 user ID, it has access to the DB2 resources directly from SPUFI, QMF™, and other utilities.

We recommend you use either the SIGN 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 may find it convenient to have just one authorization ID under which all dynamic SQL in CICS is executed.

Qualified or unqualified SQL

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

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 DB2ENTRY and DB2CONN. 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 DB2ENTRY or DB2CONN. Another possibility is to design a CICS transaction ID (using the same authorization ID) that itself could do the CREATE SYNONYM statement. However, neither of these methods is advisable.
- 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 DB2ENTRY or DB2CONN. In most cases, it must be the SIGNID, or a character string. This restriction is normally not acceptable.

For these reasons, the use of unqualified SQL in dynamic SQL statements is not recommended.

Chapter 7. Program preparation and testing

This chapter discusses program preparation and testing in a CICS DB2 environment:

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

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.

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, terminals, and DB2 definitions to be dynamically added to a running system.

In releases of CICS before CICS Transaction Server Release 2, it was not recommended to use a single CICS system for both production and test because the RCT could not be dynamically changed. Although dynamic change to the CICS DB2 definitions is now possible without stopping the attachment facility, this environment is still not recommended because applications in test could affect the performance of the production system.

Two CICS systems

Running a test CICS system and a production CICS system separately allows test failures without impacting production.

Two CICS subsystems can run with one or more 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.

Preparation steps

The steps shown in Figure 23 summarize how to prepare your program for execution after your application program design and coding is complete.

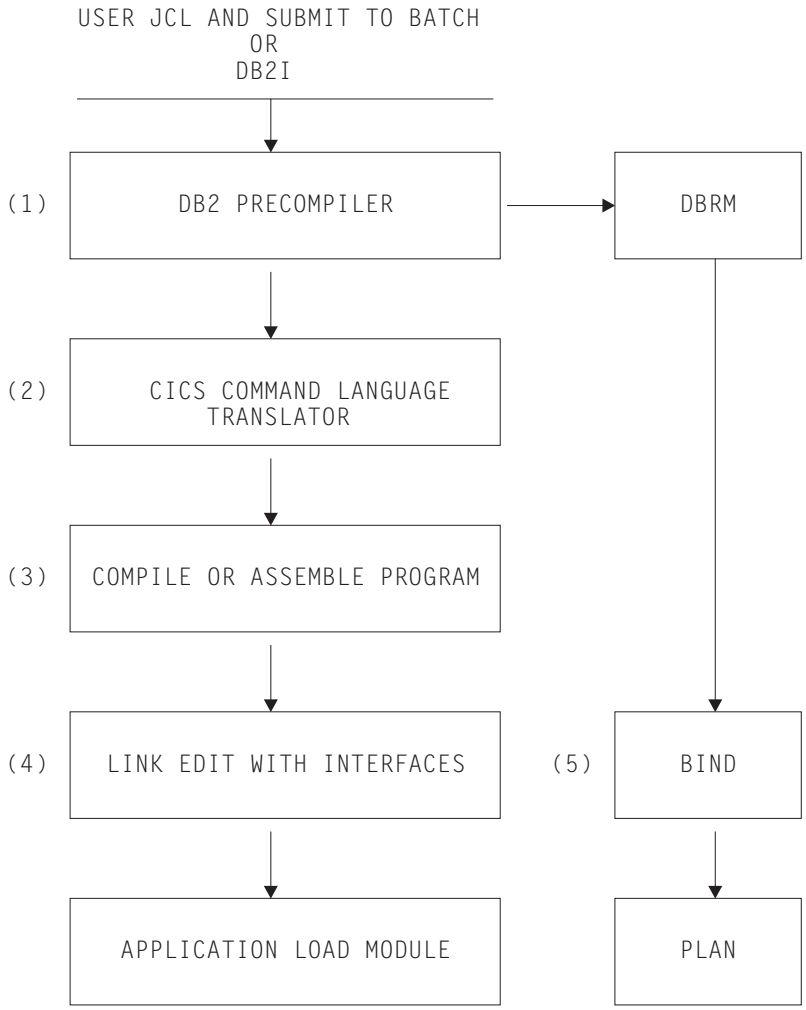


Figure 23. Steps to prepare a CICS application program

Figure 23 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. The sequence shown is that 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 DB2ENTRY 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 23 on page 66 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 the *IBM DATABASE 2 Application Programming and SQL Guide*.
- User JCL submitted to batch execution: Members DSNTEJ5C and DSNTEJ5P in the DB2 library, SDSNSAMP, contain samples of the JCL required to prepare COBOL and PL/I programs for CICS.

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.

What to bind after a program change

The example in Figure 24 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.

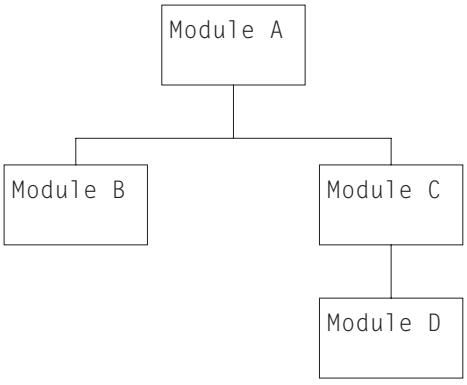


Figure 24. Application consisting of four program modules

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. Precompile the program on DB2.

2. Translate the program using the CICS translator.
3. Compile the 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 module 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 "Packages" on page 100 and the *IBM DATABASE 2 Application Programming and SQL Guide* for details of packages implementation.

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 it is recommended that you use the RETAIN option when binding your plans in the CICS environment.

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 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 the DB2 library 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 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, you should consider the possibility of using DSNTIAC instead. If you do, you must relink-edit your modules, either using another library or renaming the modules for fall back.

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.

The necessary program definitions for DSNTIAC and DSNTIA1 are provided in IBM supplied group DFHDB2 on the CSD. You must add the SDSNLOAD library to the CICS DFHRPL concatenation (after the CICS libraries) so that DSNTIA1 can be loaded.

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.

For information about these and other problem determination processes, see “Chapter 11. Problem determination” on page 121.

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
- RDO DB2ENTRY and DB2TRAN definitions for the applications. RDO DB2CONN definition if it is the first DB2-oriented application going into production

In addition, if RACF is installed, you need to define new users and DB2 objects.

Additional considerations for going into production

- *Type of threads*. When defining the new transactions and application plans in the DB2ENTRY you can use unprotected threads to get detailed accounting and performance information in the beginning. Later, you can use protected threads as needed.
- *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 Td*x* (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 Ts*x* in the SQL parameter list (for example Ts1 and Ts2, if two source modules are involved).

At BIND time, both modules in the 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 4 on page 72 and Table 5 on page 72 show one method of migration.

Table 4. 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 5. 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, it is recommended that you run EXPLAIN when you first bind the plan on the production system, to check the DB2 optimizer decisions.

Support for Java™ programs

APARs PQ34321, PQ38446
Documentation changed for PQ34321 on 20/03/00, and for PQ38446 on 17/07/00

Java programs for CICS can access DB2 using the JDBC or SQLJ application programming interfaces. This applies both to programs compiled using VisualAge for Java Enterprise Edition for OS/390 and to CICS JVM programs. For Java programs for CICS compiled using VisualAge for Java Enterprise Edition for OS/390, the following APARs are required:

- PQ34321, PQ36041, PQ37771 and PQ39969 on CICS TS 1.3
- PQ44113 on DB2 5.1
- PQ44115 on DB2 6.1
- PQ38178 on VisualAge for Java, Enterprise Edition for OS/390, Version 2.0 compiler

- PQ38179 on VisualAge for Java, Enterprise Edition for OS/390, Version 2.0 runtime (Note also that this support requires CICS to be running on OS/390 Version 2 Release 6 or higher.)

For CICS JVM programs, the following APARs are required:

- PQ34321, PQ36041, PQ37771 and PQ39969 on CICS TS 1.3
- PQ44113 on DB2 5.1
- PQ44115 on DB2 6.1

The JDBC and SQLJ requests from a Java application for CICS are processed first by the DB2 supplied type 2 JDBC driver which converts the requests into EXEC SQL equivalents. In a CICS environment, the DB2 JDBC driver is linkedited with the CICS DB2 language interface (stub) DSNCLI. Therefore, requests from the DB2 JDBC driver flow into CICS and the CICS DB2 attachment facility in exactly the same way as EXEC SQL requests from a Cobol program, for example. Hence there are no operational differences between Java programs for CICS DB2 and Cobol programs for CICS DB2, and all customization and tuning options available using the RDO equivalent of the RCT apply to Java programs for CICS DB2. Full details of how to code and build Java applications that use the SQLJ and JDBC application programming interfaces can be found in the *DB2 for OS/390 Application Programming Guide for Java Version 5 (SC26-9547-01)*, or in the *DB2 for OS/390 Application Programming Guide for Java Version 6 (SC26-9018-01).*

Particular programming features that apply to JDBC and SQLJ when used in a CICS environment are as follows:

Programming rules

The DB2 type 2 driver supports the JDBC 1.2 and the SQLJ Part 0 (ANSI 98) level of application programming interfaces, and so Java programs for CICS must adhere to the programming rules of these application programming interfaces, which are more restrictive than the general CICS programming model. See “The synconreturn environment” on page 74 below.

System properties required by the JDBC driver

Most properties in the *db2sqljjdbc.properties* file are not used in a CICS environment. See Appendix B of the *DB2 for OS/390 Application Programming Guide for Java Version 5 (SC26-9547-01)*, or the *DB2 for OS/390 Application Programming Guide for Java Version 6 (SC26-9018-01)*, for which properties are used.

Using a url to identify a data source

A JDBC or SQLJ application has to obtain a connection or connection context to a ″data source″ before executing SQL statements. The data source is identified by specifying a database Uniform Resource Locator (URL) to the JDBC driver. The basic structure of the url for DB2 UDB for OS/390 and z/OS is:

```
jdbc:db2os390:<location-name>
```
or
```
jdbc:db2os390sqlj:<location-name>
```

The location-name equates to the location name of a DB2 subsystem. Typically, this will be the location name of the local DB2 that CICS is connected to. However, a

remote DB2 can be accessed, by using the local DB2 that CICS is connected to as a pass through, and using DB2 Distributed Data Facilities to access the remote DB2.

Location-name is optional. If you omit it, then the local DB2 to which CICS is connected is used. Specifying a url of `jdbc:db2os390sqlj:` is referred to as using a **default url**. Specifying a location name in the url is referred to as using an **explicit url**.

Another supported way of specifying a default url is to specify:

```
jdbc:default:connection
```

We recommend that you use a default url in a CICS environment, for CICS applications using JDBC or SQLJ. Java applications for CICS that use JDBC or SQLJ specifying a default url do not have to close a connection on a unit of work boundary, nor is a syncpoint taken when the connection is closed (unless autocommit(true) is specified). See “The synconreturn environment” for further information. This means that there are no restrictions on interoperability between non-Java and Java applications in the same unit of work — the underlying CICS-DB2 thread is shared between the applications.

Number of connections allowed

A Java application for CICS can have at most one JDBC connection or SQLJ connection context open at a time. An application can close a connection and open a connection to a new data source (DB2 location). An application that has an open connection should close the connection before linking to another application that wishes to use JDBC or SQLJ. A commit only occurs if autocommit (true) has been specified.

Commit and rollback

JDBC and SQLJ applications are allowed to issue JDBC and SQLJ commit and rollback method calls. The DB2 JDBC driver will convert these calls into a JCICS commit or a JCICS rollback call, resulting in a CICS syncpoint being taken. Hence a JDBC or SQLJ commit results in the whole CICS unit of work being committed, not just the updates made to DB2. We do not support committing work done using a JDBC connection independently of the rest of the CICS unit of work. A JDBC or SQLJ application can also issue JCICS commit or rollback directly, with the same result — the whole unit of work is committed or rolled back together, both DB2 updates and updates to CICS controlled resources.

Autocommit

JDBC applications may use the autocommit property of a JDBC connection, which causes a commit after each update to DB2, but this commit will be a CICS commit and results in the whole unit of work being committed. Autocommit also causes a commit to be taken when a connection is closed, and applies both to connections obtained using an explicit url, and connections obtained using a default url. Use of autocommit in a CICS environment is discouraged, so the DB2 JDBC driver sets a default of autocommit(false) when running in a CICS environment, which differs from non-CICS environments where the default is autocommit(true).

The synconreturn environment

Java applications for CICS that use JDBC or SQLJ specifying an explicit url operate in an environment similar to that of a DPL server program linked to with the

SYNCONRETURN attribute, in that the closing of a connection or connection context must occur on a unit of work (UOW) boundary. Also, when the application program completes, a syncpoint must occur. For a standalone application this is not a problem, as CICS will ensure the end of task syncpoint is taken. However, the JDBC 1.2 and SQLJ application programming interfaces do not support the concept of multiple application programs for each UOW. Therefore it will not always be possible to replace an existing program in an application suite with a Java program for CICS using JDBC or SQLJ, particularly if the calling application has already accessed DB2 in the same UOW. These restrictions can be overcome by using a default url instead of a specific url (see “Using a url to identify a data source” on page 73).

CICS abends

CICS abends issued whilst processing an EXEC SQL request built by the JDBC/SQLJ driver are not converted into Java Exceptions and therefore are not catchable by a Java application for CICS. The CICS transaction will abend and rollback to the last syncpoint.

Chapter 8. Customization

The introduction of the online CICS DB2 resource control table has customization implications as described in:

- "Dynamic plan exits"

Dynamic plan exits

APAR PQ41401
Documentation changed 05 December 2000

The sample plan exit, DSNCUEXT, part of the CICS DB2 attachment facility, is integrated for CICS Transaction Server release 3 into the CICS product. Assembler source for DSNCUEXT is shipped in the SDFHSAMP library. A load module is shipped in SDFHLOAD. It is invoked as a CICS user-replaceable module.

A parameter list is passed to DSNCUEXT through a COMMAREA and has the following format in the Assembler version:

Table 6. Example of a parameter list passed to DSNCUEXT through a COMMAREA

	CPRMPLAN	DS	CL8	The DBRM/plan name of the first SQL statement on entry to DSNCUEXT, The field can be modified to establish a new plan
	CPRMAUTH	DS	CL8	The current authorization ID that is passed to DB2 at sign-on time. This is for information only. Any changes made to it are ignored.
	CPRMUSER	DS	CL4	A user area that is reserved for use by DSNCUEXT. The CICS DB2 attachment preserves this field across invocations of DSNCUEXT.
#	CPRMAPPL	DS	CL8	The name of the application program that issued the SQL call.

The Assembler version of the parameter list is shipped as member DSNCPRMA in SDFHMAC library.

The COBOL version is DSNCPRMC in SDFHCOB library.

The PL/I version is DSNCPRMP in SDFHPLI library.

Dynamic plan exits can run unchanged; they do not have to be reassembled. The parameters passed to the exits are unchanged.

A dynamic plan exit is driven to determine which plan to use at the start of the first unit of work (UOW) of the transaction. This is referred to as *dynamic plan selection*.

A dynamic plan exit can also be driven at the start of a subsequent UOW (assuming the thread was released at syncpoint) to determine what plan to use for the next UOW. The plan exit can decide to use a different plan. For this reason this is referred to as *dynamic plan switching*.

| In releases of CICS earlier than CICS Transaction Server Version 1 Release 2, dynamic plan switching could only occur for the pool, or for RCT entries that specified THRDA=0, that is, overflowed to the pool. A consequence of using DSNC MODIFY to modify THRDA to zero was that dynamic plan switching became

effective! An RCTE with THRDA > 0 would not be capable of dynamic plan switching; the plan selected for the first UOW would be used for all subsequent UOWs of the transaction.

In CICS Transaction Server for OS/390 Release 3 dynamic plan switching can occur for both DB2 entries as well as the pool, irrespective of the THREADLIMIT parameter. If you have coded your own dynamic plan exit, check that the logic copes with subsequent invocations for the same task. Either the user application or the dynamic plan exit must be written to tolerate consequences of additional calls to the exit. If the dynamic plan exit would change the plan when not wanted, the user application can avoid this by ensuring the thread is not released at syncpoint. Preferably, if the thread is released, the dynamic plan exit must provide the proper plan for the new cases when it is called, that is, a DB2ENTRY with THREADLIMIT > 0.

Before calling the dynamic plan exit, the CICS DB2 attachment facility sets CPRMPLAN to the name of the DBRM set in the parameter list of the first EXEC SQL statement executed in the unit of work. The shipped default dynamic plan exit DSNCUEXT does not modify the plan name as input in CPRMPLAN by the CICS DB2 attachment facility, but returns immediately, leaving the plan name as that chosen by the CICS DB2 attachment facility.

As a consequence of adding support for JDBC, and SQLJ support for CICS Java Applications (by applying APAR PQ34321), the shipped default dynamic plan exit has been changed. SQLJ and JDBC require that DB2 produce four DBRMs for each application program in order to support dynamic change of isolation levels. For JDBC and SQLJ applications, the DBRM name is restricted to seven characters, the eighth character being used as a suffix of 1,2,3 or 4. Hence for JDBC and SQLJ applications it is not possible to use a default naming convention of program name = dbrm name = plan name, as the DBRM name being used will contain a suffix of 1,2,3 or 4.

APARs PQ34321 & PQ38446

Documentation changed for PQ34321 on 20/03/00 & for PQ38446 on 17/07/00

For JDBC applications, SQLJ applications, and mixed JDBC and SQLJ applications, the DB2 JDBC driver uses information from the JDBC profile to set the name of the DBRM in the parameter list of the first EXEC SQL statement executed. The first SQL issued will always have the DBRM name set to the JDBC base program with the default isolation level appended. That is, DSNJDBC2 by default, or if, for example, the JDBC profile was generated using *pgmname=OTHER*, the DBRM name will be OTHER2. For example, if dynamic plan exits are not used, the plan name is always obtained from the DB2CONN or DB2ENTRY definition, the plan name in the properties file is ignored.

In order to support a default naming convention for JDBC and SQLJ, the shipped default sample dynamic plan exit DSNCUEXT has been changed to detect an input CPRMPLAN name whose first seven characters are 'DSNJDBC' (or 'DSNSQLJ'). If such a plan name is detected, the plan name is changed to DSNJDBC', (with the eighth character set to blanks). Users wishing to use the default dynamic plan exit with CICS Java applications should bind the multiple DBRMs into a plan called DSNJDBC.

Chapter 9. Accounting

This chapter discusses accounting data delivered by CICS and DB2 in the following sections:

Overview

Accounting in a CICS environment can be used 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 transaction accessing a 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.

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.

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 two classes of data:
– Performance class for detailed transaction level information
– Exception class for exceptional conditions

For a detailed description of the CICS monitoring facilities, see the *CICS Performance Guide*, and the *CICS Customization 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. The TIVOLI™ Performance Reporter for OS/390 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 Customization Guide* show how to manipulate the SMF records. The TIVOLI Performance Reporter can build reports that help you with:
- Systems overview
- Service levels
- Availability
- Performance and tuning
- Capacity planning
- Change and problem management
- Accounting

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 DB2CONN or DB2ENTRY). 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 (PROTECTNUM>0) and all transactions with the same transaction code for this DB2ENTRY 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 set ACCOUNTREC in your DB2ENTRY or DB2CONN definitions to UOW, TASK, or TXID. Setting ACCOUNTREC to these options is considered a sign-on, even if you use the same authorization ID.

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.

Accounting for processor usage

This section provides information on the reporting of processor resources used by CICS and DB2. Details about the different accounting classes in the DB2 accounting record are also discussed.

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.

For example, data in the dispatcher statistics section provides the accumulated time for each of the CICS TCBs. The field Accum/TCB is the total processor time used by the corresponding TCB. For more information about the CICS Dispatcher statistics, see the *CICS Performance Guide*

You can obtain the total processor time used by the CICS address space from RMF workload (WKLD) reports. This time is usually 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
- 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 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 threads always use MVS subtasks separate from the CICS main task TCB. Consequently, you need more granularity to get the proper results.

You can obtain transaction performance records from reports generated by TIVOLI Performance Reporter for OS/390. TIVOLI reads the SMF datasets searching for SMF type 110 records, in the case of CICS, and provides a matrix, tabular or any graphics for each transaction run in the CICS system. For more information about the TIVOLI Performance Reporter for OS/390, see the*CICS Performance Guide*

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 the *DB2 for OS/390: 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.

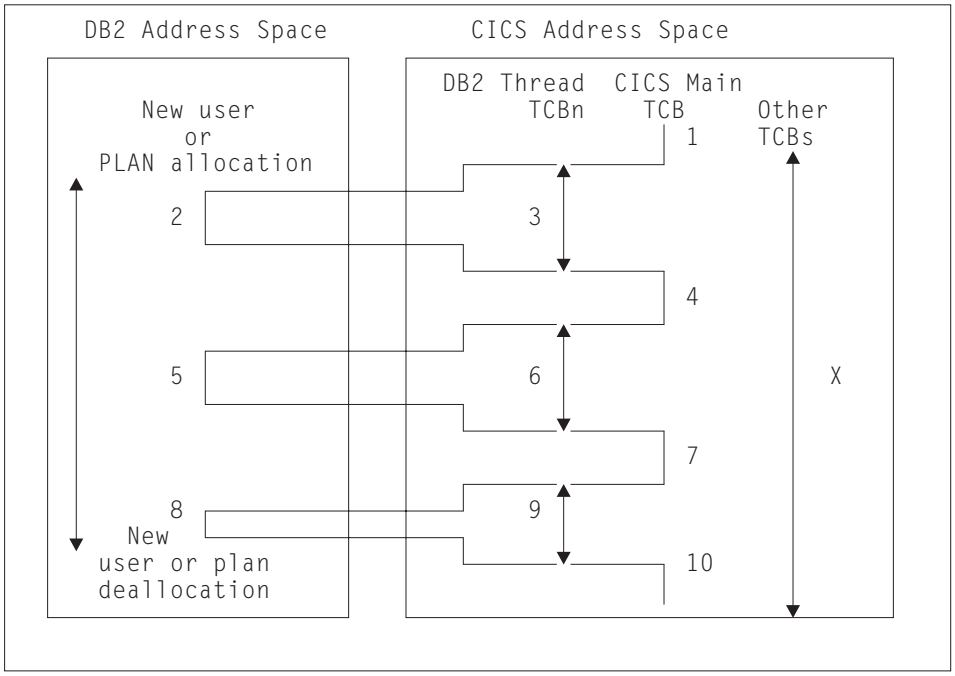
In a CICS environment, the processor time from the DB2 accounting records is typically much greater than the processor time reported in the DB2 statistical records, because most of the processor time used is in the thread TCB itself and in the DB2 address spaces using cross memory services.

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.

You can find a description of the contents of the DB2 accounting record in the *DB2 for OS/390: Administration Guide*. There is also the description of accounting record fields in member DSNWMSGS, which is shipped in SDSNSAMP.

Figure 25 on page 84 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. You can use the ACCOUNTREC(TXID) parameter in the DB2ENTRY or DB2CONN 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.



CLASS 1 reported processor time = 3 + 6 + 9
CLASS 2 reported processor time = 2 + 5 + 8
CICS reported processor time = 1 + 4 + 7 + 10

Figure 25. 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 ACCOUNTREC(TASK) or ACCOUNTREC(UOW) settings in a DB2ENTRY or DB2CONN provide more granularity, suppress some strange records, and allow 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.

ACCOUNTREC(TASK) ensures that there is a minimum of one accounting record for each task. There could be more, depending on thread reuse. ACCOUNTREC(TASK) is recommended rather than ACCOUNTREC(UOW).

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 rollback), the DB2 thread associated with the transaction is released after each syncpoint. (POOL threads are generally terminated, while DB2ENTRY 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 is 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.

Non-terminal oriented transactions may or may not release the thread at syncpoint, depending on the NONTERMREL parameter in the DB2CONN. If NONTERMREL(YES) is set, the thread is released at syncpoint, and the considerations described apply. If NONTERMREL(NO) is set a non-terminal-oriented transaction does not release the thread at syncpoint. That means that only one DB2 accounting record is produced for this kind of transaction. However, if other transactions are specified for the same DB2 entry, the DB2 accounting record for this thread can include data for these transactions as well, if the thread is reused.

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 the *DB2 for OS/390: 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 25 on page 84, and is the sum of the times 2, 5, and 8. Note that in the case of CICS (in contrast to IMS™ and TSO), this is not significantly different from the CLASS 1 trace information.

Terminal-driven transactions (and non-terminal-driven transactions if NONTERMREL=YES) 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 plans and locations of specific authorization ID's The processor overhead is 0-5%, with 2% being typical.

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 asynchronous VSAM.

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.
- If the CLASS 2 processor time is subtracted from the CLASS 1 processor time, this gives an approximation of CPU utilization of the CICS attachment facility. This is a useful tuning aid.
- 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.

For an explanation of capture ratio, see the *CICS Performance Guide*. 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.

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 required accounting level. “Availability of accounting data” on page 88 discusses under which conditions this data is available (the second activity).

Types of accounting data

The following data types, which could be used for accounting, are discussed in subsequent sections:

- Processor usage
- I/O
- GETPAGE
- Write intents
- SQL call activity
- Transaction occurrence
- Storage.

Processor usage

The processor usage information given in the DB2 accounting record shows 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 “Accounting for processor usage” on page 81 for more detail on reporting processor usage in a CICS DB2 environment.

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.

GETPAGE

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.

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.

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.

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.

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.

Availability of accounting data

When you decide which resources to account for, you must find a 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.

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 DB2ENTRY or DB2CONN parameter AUTHTYPE as OPID, 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 “Plan execution authorization” on page 61, 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 (OPID, 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 OPID, USERID, GROUP, and TERM is not an attractive solution.

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 ACCOUNTREC(UOW) or ACCOUNTREC(TASK) in the DB2ENTRY or DB2CONN, CICS passes its LU6.2 token to DB2 to be included in the DB2 trace records. The token is written to QWHCTOKN in the correlation header and simplifies the task of correlating CICS and DB2 accounting records. If you do not use ACCOUNTREC(UOW) or ACCOUNTREC(TASK), there is no easy way to do this matching. Two problems exist:

1. 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.
2. No fields in the DB2 accounting record can be used to identify exactly the corresponding CICS performance records. This is because there is no one-to-one relationship between the two record types.

Figure 26 on page 90 shows what you can do when you are not using ACCOUNTREC(UOW) or ACCOUNTREC(TASK) 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
- The authorization ID field
- The CICS LU6.2 token

Note that there is no ideal way of combining the two record types. There may be cases where it is impossible to make the matching correct because the transactions are run concurrently.

Figure 26 shows four combinations of using transaction codes and the number of transactions reported in one DB2 accounting record.

	One transaction in each DB2 accounting record	Many transactions in each DB2 accounting record
Each transaction path has its own transaction ID	A	B
Many different transaction paths share the same transaction ID	C	D

Figure 26. Different accounting cases

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.

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. ACCOUNTREC (TASK) ensures that there is a minimum of one accounting record for each task. There could be more, depending on thread reuse. ACCOUNTREC(TASK) is recommended rather than ACCOUNTREC(UOW).

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 ACCOUNTREC(UOW) or ACCOUNTREC(TASK). In cases B and D, the thread contains information from many transactions.

Four accounting cases

The four different accounting cases, Case A through Case D, are discussed in more detail below.

Case A

There is at least one DB2 accounting record and at least one CICS performance record for a transaction making 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 AUTHID/AUTHTYPE parameter in the DB2CONN or DB2ENTRY.

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 units of work 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 at syncpoint.

The same applies to non-terminal-oriented transactions if NONTERMREL(YES) is set in the DB2CONN, meaning that the thread is released at syncpoint.

If two or more different transactions use the same DB2ENTRY, 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 record 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 records can be used to distinguish one transaction from another. For example, the user could supply information in a user field in the CICS performance records. This information could define the transaction being executed. The model transactions can then be used.

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 are 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:
  1. Running all DB2 activity under the authorization ID of the end user. This is obtained by setting the AUTHTYPE parameter in the DB2CONN or DB2ENTRY to either OPID, 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 described in the previous sections.
  2. 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 ACCOUNTREC(UOW) or ACCOUNTREC(TASK) parameter on the DB2CONN or DB2ENTRY definition to pass the CICS LU6.2 token to DB2 for correlation purposes. This carries 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 DB2CONN/DB2ENTRY 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 DB2CONN and DB2ENTRY with respect to the trade-off between performance and accounting requirements.

Chapter 10. 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 discusses CICS DB2 application design considerations in the following sections:

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.

See the *IBM DATABASE 2 Application Programming and SQL Guide* for information on DB2 application design and the *CICS Application Programming Guide* for information on CICS application design.

Overview

Using DB2 as the data base management system (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 development 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.

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 DB2CONN or DB2ENTRY 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.

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 and some consequences for the two types of SQL statements are shown in Table 7 on page 97.

Table 7. Qualified and unqualified SQL

Relationship to other DB2 areas	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

Some of the limitations shown in Table 7 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.

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. DB2 also provides the ability to lock at the row level rather than the page or tablespace level, thus providing better granularity and reducing lock contention.

In general, it is recommended that you design 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).

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.

The following sections describe some specific recommendations for BIND options with CICS:

Isolation level

It is recommended that 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 together to use the same DB2ENTRY and let them use a common plan, then this new plan must also use RR, if just one of the transactions requires RR.

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 work (UOW).

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 syncpoint. It is not recommended that you use this option with 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.

ACQUIRE and RELEASE

The general recommendations for these parameters are described in “Coordinating DB2CONN, DB2ENTRY, and BIND options” on page 41. The parameters change from plan to plan and over time, because they are related to the transaction rate. The following section is under review

Using protected threads

Using protected threads is a performance option that reduces the resources involved in creating and terminating a thread. For performance reasons, it is recommended that you use protected threads whenever possible. See “Chapter 5. Defining the CICS DB2 connection” on page 37 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. This can be overcome by specifying ACCOUNTREC(UOW) to make sure an accounting record is cut per unit of work, or by specifying ACCOUNTREC(TASK) to make sure there is an accounting record cut per CICS task. See “Chapter 9. Accounting” on page 79 for more information about accounting in a CICS DB2 environment.

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, the CICS DB2 attachment 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 6. Security” on page 49.

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 specified in:

- DB2ENTRY, exit for a specific transaction code specified in the keyword PLANEXITNAME
- DB2CONN, exit for transactions using the pool specified in the keyword PLANEXITNAME.

IBM supplies a sample exit program, DSNCUEXT, in assembler language and object code. You can also write other exit programs.

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 (unless program autoinstall is being used.
- 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.

Sample exit program

The object code for sample, DSNCUEXT, is included in the SDFHLOAD library. The supplied source code is written in assembler language and supplied in the SDFHSAMP library. The sample program shows how to address the parameter list but does not change the plan name.

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. The dynamic plan exit was designed to select a plan dynamically at the start of the transaction (dynamic plan selection) not to change plans frequently within transactions.

To do dynamic plan switching the thread must be released at syncpoint and reacquired to drive the dynamic plan exit. In releases of CICS before CICS Transaction Server Version 1 Release 2, dynamic plan switching occurred only for pool threads. After that time switching occurs also for DB2ENTRY threads.

To invoke the dynamic plan exit to do plan switching after the end of a UOW your transaction must release the thread at syncpoint. A transaction releases a thread at syncpoint only if:

- It is a terminal driven task, or a nonterminal driven task and NONTERMREL=YES is set in the DB2CONN
- No held cursors are open
- Any DB2 special registers modified have been set back the their initial state.
- DB2 special register CURRENT DEGREE has ever been modified by this transaction.

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 defnitions. 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 by 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, and 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 DB2ENTRY definitions.

- Reduce CICS STARTS or exits.
- Avoid cloning CICS and DB2ENTRY definitions.
- Provide the ability to bind a plan with null packages for later inclusion in the plan.
- Allow you to specify collection at execution time using 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

There are other benefits that using packages can provide in a CICS environment:

- It allows you to use fewer plans.
- It allows you to bind low-use transactions into a single plan.
- It 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 the *IBM DATABASE 2 Application Programming and SQL Guide* and *DB2 Packages: Implementation and Use*.

Avoiding AEY9 abends

You can use the following CICS command to detect whether the CICS attachment facility is enabled:

```
EXEC CICS EXTRACT EXIT PROGRAM('DFHD2EX1')
         ENTRY('DSNCSQL')
         GASET(name1)
         GALENGTH(name2)
```

If you specify a program name of DSNCEXT1 or DSN2EXT1 CICS dynamically changes it to the required name DFHD2EX1. If you get the INVEXITREQ condition, the CICS attachment facility is not enabled.

When the CICS attachment facility is enabled it is not necessarily connected to DB2. It can be waiting for DB2 to initialize. When this occurs, and an application issues an EXEC SQL command when CONNECTERROR=ABEND is specified in the DB2CONN, an AEY9 abend would result. CONNECTERROR=SQLCODE would result in a -923 SQL code being returned to the application.

You can use the INQUIRE EXITPROGRAM command with the CONNECTST keyword in place of the EXTRACT EXIT command to determine whether the CICS is connected to DB2.

The CONNECTST keyword of the INQUIRE EXITPROGRAM command returns values:

- 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.

If the command fails with PGMIDERR, this is the same as NOTCONNECTED.

Figure 27 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'DFHD2EX1'
         EXEC CICS INQUIRE EXITPROGRAM(EXITPROG)               X
               ENTRYNAME(ENTNAME) CONNECTST(CSTAT) NOHANDLE
         CLC   EIBRESP,DFHRESP(NORMAL)
         BNE   NOTREADY
         CLC   CSTAT,DFHVALUE(CONNECTED)
         BNE   NOTREADY
```

Figure 27. Example of the INQUIRE EXITPROGRAM command

If you specify a program name of DSN2EXT1, CICS dynamically changes it to the required name, DFHD2EX1.

Further consideration on the use of the EXTRACT EXIT or INQUIRE EXITPROGRAM commands by applications has to be made when running in an environment where dynamic workload balancing using the MVS workload manager is taking place.

If an application avoids making DB2 calls because it knows the CICS DB2 connection is not active, but issues an error message instead and returns normally, it could delude the workload manager into routing more work to the CICS region. This is called the 'storm drain effect'. Because the application did not abend, the workload manager believes that good response times are being achieved by this CICS region for DB2 work, and routes more work down the 'storm drain'.

This effect can be avoided by doing either of the following:

- Ensuring the application abends.
- Running with STANDBYMODE=RECONNECT and CONNECTERROR=SQLCODE in the DB2CONN. Applications should not use the Extract or Inquire Exitprogram commands to test whether DB2 work is possible. Instead, a test should be made for -923 SQLcode to be returned if CICS is not connected to DB2. At the time of returning the -923 sqlcode, the CICS DB2 attachment facility informs the MVS workload manager that the request has failed, and the 'storm drain effect' is avoided.

You are, therefore, advised to do the following:

- Specify STANDBYMODE=RECONNECT in the DB2CONN. This ensures that the CICS DB2 attachment facility waits (in standby mode) for DB2 to initialize and connect automatically, should DB2 be down when connection is first attempted. Also, if DB2 subsequently fails, the CICS DB2 attachment facility reverts again to standby mode and wait for DB2. It then automatically connects when DB2 returns.

- Use CONNECTERROR=SQLCODE provided applications handle the -923 code correctly.
- Avoid using EXTRACT EXIT or INQUIRE EXITPROGRAM commands if CONNECTERROR=SQLCODE can be used.
- Use CONNECTERROR=ABEND if an AEY9 abend is required. Use the INQUIRE EXITPROGRAM command instead of the EXTRACT EXIT command.
- It is worth noting that AEY9 abends can still occur even when STANDBYMODE=RECONNECT and CONNECTERROR=SQLCODE are specified if:
  - The CICS DB2 attachment facility is never started. An AEY9 results if an application issues an EXEC SQL command. You should always specify DB2CONN=YES in the SIT, or program DFHD2CM0 in PLTPI. Therefore the CICS DB2 attachment is at minimum in standby mode.
  - The CICS DB2 attachment is shut down using a DSNC STOP or CEMT/EXEC CICS SET DB2CONN NOTCONNECTED command.

It is advisable to avoid shutting down the attachment. The CICS DB2 SPI commands allow dynamic modification of the environment without shutting down the attachment.

CICS and CURSOR WITH HOLD option

The WITH HOLD option on a CURSOR declaration in a CICS program causes the following effects during a 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 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. The precompiler cannot detect 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. 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.

EXEC CICS RETURN IMMEDIATE command

When the TRANSID option is specified in conjunction with the 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 advisable, for example, to keep DB2 plan sizes smaller or to have better accounting statistics for charge-back purposes.

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 DB2CONN or a DB2ENTRY. The plan can be named explicitly, or a plan exit routine can be named that selects the plan name. 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 “CICS DB2 design criteria” on page 95.

A sample application

A simple example can be used to explain the consequences of different application design techniques. Figure 28 shows how CICS MAPs and transaction IDs are correlated, and how the transactions should work, without DB2 considerations. In this example:

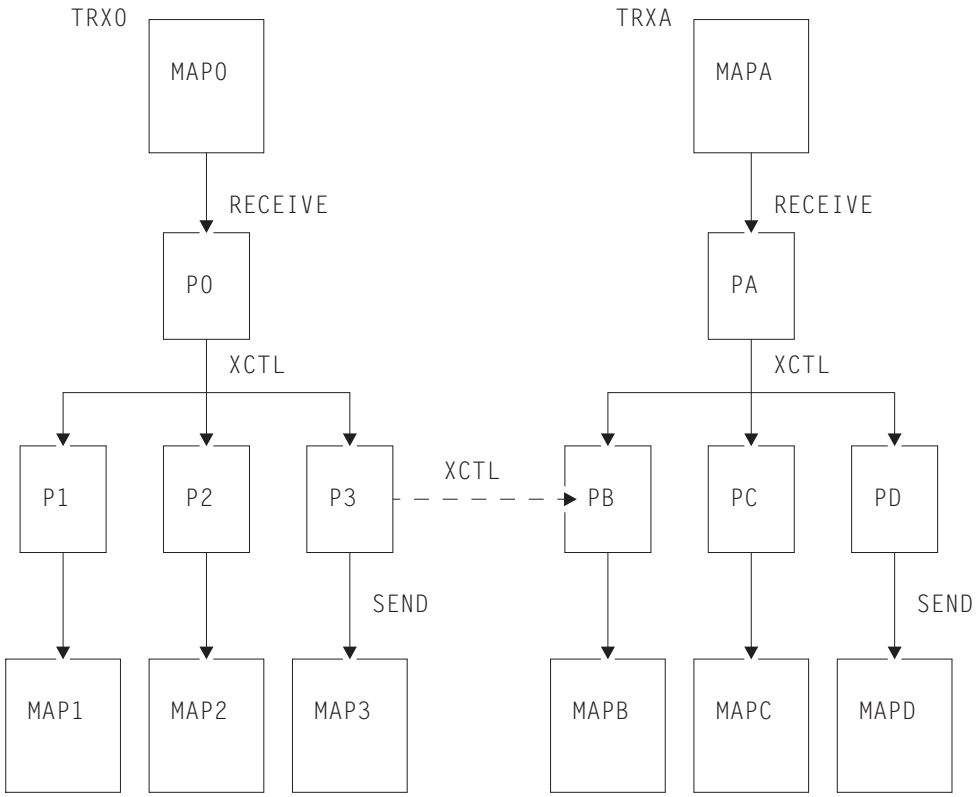


Figure 28. Example of a typical application design

- 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.

But first the general problem with plan switching is discussed.

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. It is strongly recommended 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 UOW. If the new task is backed out, the work done in the first task remains committed.

One large plan

Using one large plan for all CICS transactions that issue SQL calls is an easy strategy. For the example in Figure 28 on page 105:

- There is one plan, PLAN0, using the DBRMs from P0, P1, P2, P3, PA, PB, PC, and PD
- In CICS, define one DB2ENTRY specifying PLAN0 plus a DB2TRAN per transaction ID required (unless a wildcard transaction ID can be specified). One DB2ENTRY gives the best overall thread utilization if protected threads are used.

Advantages

- There are 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 definition in CICS.
- Each DBRM exists in only one plan (PLAN0).
- One large plan is 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 DB2ENTRY.

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 the *IBM DATABASE 2 System Monitoring and Tuning Guide*.

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 28 on page 105, 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 “Switching CICS transaction codes” on page 106.

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.

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.

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.

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 28 on page 105 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 "Switching CICS transaction codes" on page 106. It is recommended 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.

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.

Disadvantages

- Plan overlap can occur.
- The optimum grouping can change over time.
- Plan switching may be necessary.
- Plan switching is handled in the application code.

A fallback solution

You can use this technique if the applications were developed with little attention to the DB2 plan aspects. After the application is completely developed, the plans are defined to match the transaction.

In general this technique is not recommended, 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 DB2ENTRY 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 is used under more than one CICS transaction code.

   The output from this step is a list of DBRMs for each transaction.

2. For each CICS transaction code decide which plan it should use. (Only one plan may be specified in the DB2ENTRY 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.

Applied to the example in Figure 28 on page 105, 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
- Two DB2ENTRY definitions, one for each plan

The advantages and disadvantages of the fallback technique completely depend on the actual application design and the result of the above-mentioned steps.

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.

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: It is recommended that you use packages rather than this technique to control program flow.

Figure 29 on page 111 shows an example of the contents in the control table. The example is based on the design situations described in Figure 28 on page 105.

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 PFCP program can accept either a function name or a program name.

PFCP then uses the function column to search for a valid transaction.

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 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 using the DB2ENTRY and DB2TRAN CICS definitions

When implementing the definitions in CICS, you should consider the following:

- Previously, many different macro RCTs could be used by a CICS system over a period of time. Therefore, different RCT names could be used at different times.
- The same RCT name can be used by different CICS systems.
- With RDO-defined CICS DB2 definitions, the need for multiple RCTs is removed, because DB2ENTRYs and DB2TRANS can be installed or discarded as required using the same DB2CONN.

Multiple 'logical RCTs' could be maintained in an RDO environment by placing a DB2CONN definition and its associated DB2ENTRYs and DB2TRANs in a list, and installing the required list. Two additional columns would be added to the control table:

1. DB2CONN name
2. CICS APPLID

At execution time, the PFCP can determine the applid and DB2CONN name using an EXEC CICS INQUIRE SYSTEM command. It then searches for rows belonging to the current applid and DB2CONN.

The disadvantage, however of multiple 'logical RCTs' is that a DB2CONN cannot be discarded and replaced without shutting down the CICS DB2 attachment facility. A solution is to have one DB2CONN definition and to change 'RCTs' by discarding and installing different sets of DB2ENTRYs and DB2TRANS to work with it. These can be discarded and installed without shutting down the attachment facility.

In this case, the control table could contain a DB2ENTRY name instead of the DB2CONN. At execution time, the PFCP can use the CICS DB2 SPI commands to inquire whether this DB2ENTRY was installed. In this way using the name of one DB2ENTRY to identify the set of definitions installed, identifies the 'logical RCT'.

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	*

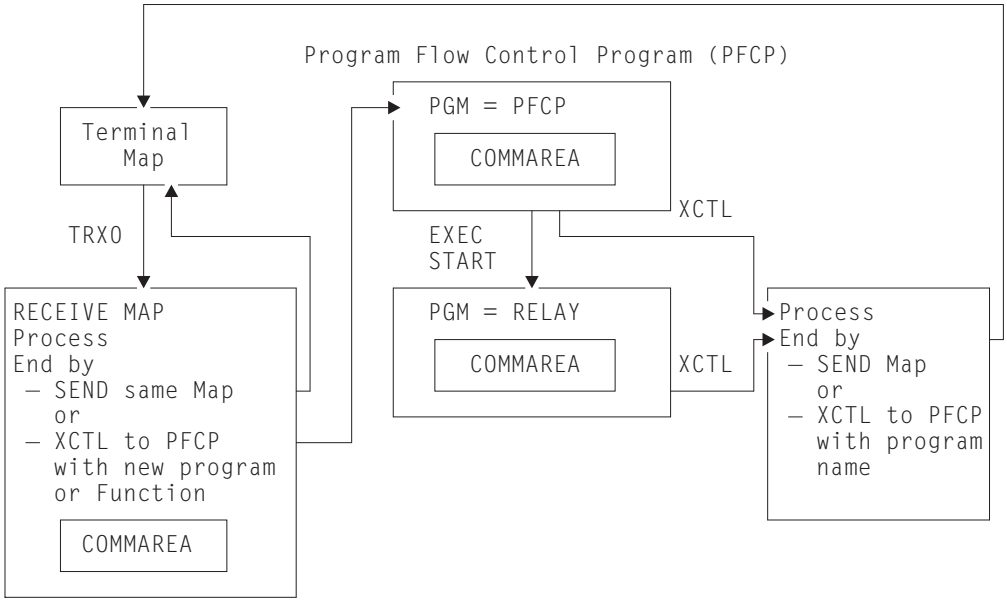


Figure 29. Table-controlled program flow

Program flow

The principle in the program flow is shown in Figure 29. The flow for the application design used in Figure 28 on page 105 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 (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 appropriate 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 29 on page 111.
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.

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 DB2ENTRYs and DB2TRANs in CICS, (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.

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.

Using packages

As explained in “Implementation” on page 100, the use of dynamic plan switching 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. 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 DB2ENTRY 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 doing it.

DB2 Version 2 release 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 Version 3 release 1, where you can have accounting at the package level.

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 DB2ENTRYs and DB2TRANs 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 DB2ENTRY, replace the dynamic plan switching exit program name with the name of this new plan. For example, replace:

     ```
     PLANEXITNAME=DSNCUEXT
     ```

     with

     ```
     PLAN=PLANAPP1
     ```

- One plan per transaction

  The steps in using this approach are:

  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 as 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 DB2ENTRY definitions to replace the DPS exit with the plan names associated with the transactions.

  This approach is preferable when using DB2 Version 2 release 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.

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.

Programming

This section describes the following programming considerations:

- SQL language
- Views
- Updating index columns
- Commit processing
- Serializing transactions
- Page contention

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 the *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, it is recommended 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.

Views

It is generally recommended 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.

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.

Dependency of unique indexes

A programmer can take advantage of the fact that DB2 returns only 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.

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 SYNCPOINT:

- The UOW 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, unless NONTERMREL=YES is specified in the DB2CONN. It first happens when the transaction is finished.
- 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.

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 syncpoint (for terminal-oriented transactions, and nonterminal-oriented transactions if NONTERMREL=YES is defined), you can use your DB2ENTRY 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*. It is recommended 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 syncpoint 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.

Page contention

When designing applications and databases, consider the impact of having many transactions accessing the same part of a table space. The term “hot spot” is 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 syncpoint. 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.

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 can be used. If the number of pages in the table space is low, the index is not 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.

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 “Handling deadlocks” on page 151.

Chapter 11. Problem determination

Problem determination for CICS DB2 is discussed in this chapter under the following headings:

This chapter contains Diagnosis, Modification or Tuning information.

Wait types

The CICS DB2 task related user exit, DFHD2EX1, uses the WAIT_MVS function of CICS dispatcher domain to put the running CICS task into a CICS dispatcher wait. DFHD2EX1 issues WAIT_MVS calls for different purposes and these are distinguished by different resource name and resource type values. The resource name and resource type values are visible in the dispatcher section of a CICS system dump and are also externalised on the CEMT INQUIRE TASK panel as Hty and Hva values respectively.

For example, when a CICS system dump is taken a CICS task is waiting for the CICS DB2 subtask to complete its work in DB2, this wait is represented by a resource type of 'DB2' and a resource name of 'LOT_ECB'. The examples shown in Figure 30, Figure 31 on page 122, and Figure 32 on page 122 show how this would appear in the dispatcher section of CICS system dump, and using the CEMT inquire task panels.

```
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        LOT_ECB  MVS  18:51:22
01040031 02BA9780 NS RUN
01060009 02B13B00 NS SUS Y OK -                       MVS  18:50:29
```

Figure 30. CICS-formatted dump: dispatcher domain

```
INQUIRE TASK
STATUS:  RESULTS - OVERTYPE TO MODIFY
  Tas(0000151) Tra(DSNC)           Sus Tas Pri( 255 )
? Tas(0000161) Tra(XC05) Fac(1303) Sus Ter Pri( 001 )
  Tas(0000162) Tra(CEMT) Fac(1302) Run Ter Pri( 255 )
```

Figure 31. CICS CEMT INQUIRE command for a CICS DB2 wait transaction

```
INQUIRE TASK
SYNTAX OF SET COMMAND
 Tas(0000161) Tra(XC05) Fac(1303) Sus Ter Pri( 001 )
    Hty(DB2     ) Hva(LOT_ECB ) Hti(000018) Sta(TO)
    Use(SYSADM  ) Rec(X'A8B5FE112D54CA85')
 CEMT  Set TAsk() | < All >
  < PRiority() >
  < PUrge | FOrcepurge >
```

Figure 32. CICS CEMT INQUIRE command after question mark (?)

The full list of waits issued from the CICS DB2 task related user exit DFHD2EX1, their meanings, and whether the task can be purged or forcepurged out of this wait is given in Table 8.

Table 8. WAITs issued from the CICS DB2 TRUE

Resource type or Hty value	Resource name or Hva value	Meaning	Purge and forcepurge
DB2	LOT_ECB	CICS task is waiting for DB2, that is, waiting for the CICS DB2 subtask to complete the request.	Purge — No Forcepurge — Yes (but see b
CDB2RDYQ	name of DB2ENTRY or *POOL	CICS task is waiting for a thread to become available. The resource name details for which DB2ENTRY or the pool has a shortage of threads.	No
CDB2TCB		CICS task is waiting for a CICS DB2 subtask to become available. This indicates that TCBLIMIT has been reached and all TCBs are in use running threads.	No

DB2 indicates that the task is waiting for DB2 or, more specifically, the task is waiting for its associated subtask to complete the DB2 request and post it back. The task can be forcepurged when in this state, in which case the CICS task abends and backout occurs. In addition, the CICS DB2 subtask is detached causing DB2 to back out as well. Forcepurging a task in this state can cause termination of the DB2 subsystem, if the CICS-DB2 subtask is terminated during a 'must complete' activity in DB2. To avoid this risk, use the DB2 CANCEL THREAD command before issuing the forcepurge — see "Purging CICS DB2 transactions" on page 19 for the procedure.

CDB2RDYQ indicates that the THREADLIMIT value of a DB2ENTRY or the pool has been exceeded and that THREADWAIT is set to YES indicating that the task should wait. Purge of the task in this state is not allowed. An attempt to forcepurge the task results in message DFHAP0604 being issued to the console. Forcepurge is deferred until a thread has been acquired and the task is no longer queued waiting

for a thread. Instead, a SET DB2ENTRY() THREADLIMIT(*n*) command can be used to increase the number of threads available for the DB2ENTRY, or a SET DB2CONN THREADLIMIT(*n*) if it is the pool. CICS posts tasks to try again to acquire a thread if the threadlimit is increased.

CDB2TCB indicates that the task is allowed a thread but that all CICS DB2 subtasks are in use and the TCBLIMIT specified in the DB2CONN has been reached so no new TCBs can be attached. Purge of the task is not allowed. An attempt to forcepurge the task results in message DFHAP0604 being issued to the console and forcepurge is deferred until a TCB has been acquired and the task is no longer queued waiting for a TCB. Instead a SET DB2CONN TCBLIMIT command can be used to increase the number of TCBs allowed. CICS posts tasks to try again to acquire a DB2 subtask if the TCBLIMIT is increased.

Table 9 gives details of WAITS issued using WAIT_OLDC dispatcher calls where the ECB is hand posted:

Table 9. WAITS issued using WAIT_OLDC dispatcher calls

Resource type or Hty value	Resource name or Hva value	Meaning	Purge and forcepurge
DB2_INIT		CICS DB2 initialization program DFHD2IN1 issues the wait to wait for the CICS initialization task running program DFHD2IN2 to complete.	Yes
DB2CDISC	name of DB2CONN	A SET DB2CONN NOTCONNECTED command has been issued with the WAIT or FORCE option. DFHD2TM waits for the count of tasks using DB2 to reach zero.	Yes
DB2EDISA	name of DB2ENTRY	A SET DB2ENTRY DISABLED command has been issued with the WAIT or FORCE option. DFHD2TM waits for the count of tasks using the DB2ENTRY to reach zero.	Yes

Table 10 shows details of EXEC CICS WAIT EXTERNAL requests issued by the CICS DB2 attachment facility.

Table 10. EXEC CICS WAIT EXTERNAL requests issued by the attachment facility

Resource type or Hty value	Resource name or Hva value	Meaning	Purge and forcepurge
USERWAIT	CDB2TIME	The CICS DB2 service task program DFHD2EX2 is in its timer wait cycle either waiting for the protected thread purge cycle to pop or to be posted for another event.	Yes
USERWAIT	DB2START	The CICS DB2 service program DFHD2EX2 is waiting for DB2 to post it when it becomes active.	Yes

Table 11 on page 124 gives details of EXEC CICS WAIT EVENT requests issued by the CICS DB2 attachment facility:

Table 11. EXEC CICS WAIT EVENT requests

Module	Resource name	Meaning
DFHD2EX2	PROTTERM	A TERM call has been issued for a protected thread during the protected thread purge cycle
DFHD2STR	ATCHMSUB	The CICS DB2 startup program DFHD2STR has attached the master subtask program DFHD2MSB and is waiting for it to identify to DB2
DFHD2STR	DTCHMSUB	The CICS DB2 startup program is waiting for the master subtask program DFHD2MSB to terminate.
DFHD2STP	MSBRETRN	The CICS DB2 shutdown program is waiting for the master subtask program DFHD2MSB to terminate.
DFHD2STP	CEX2TERM	The CICS DB2 shutdown program is waiting for the CICS DB2 service task CEX2 running program DFHD2EX2 to terminate all subtasks and terminate itself.

Messages

Messages issued by the CICS attachment facility use the DFHDB prefix which is also used for CICS DBCTL messages. Message numbers are in the range 2000 through 2999 and are reserved for CICS DB2. Thus CICS DB2 attachment messages are of the form DFHDB2xxx.

Where possible, message numbers have been maintained from previous releases of the attachment. For example, message DFHDB2023 documents the same condition as old messages DSN2023 or DSNC023. However, the contents of the message have changed. For example, all messages contain the date, time, and applid.

The destination for transient data messages output by the CICS DB2 attachment facility is controlled using the MSGQUEUE1, MSGQUEUE2 and MSGQUEUE3 parameters of the DB2CONN definition. The same message can be routed to three transient data queues. By default, messages are sent to a single queue called CDB2 which is defined as indirect to queue CSSL.

All messages are documented in the *CICS Messages and Codes*, manual and are available using the CMAC CICS-supplied transaction.

Trace

In releases of CICS before CICS Transaction Server Release 2, the user could specify the trace points to be used by the CICS attachment facility using parameters in the RCT. Trace points can no longer be user specified. The CICS attachment facility uses AP domain trace points in the range 3100 to 33FF. The contents of all trace points issued by the CICS attachment facility is documented in the *CICS User's Handbook*.

Standard tracing performed by the CICS attachment facility is controlled by the FC (File Control) and RI (RMI) trace flags. RMI trace controls trace output from the CICS DB2 task related user exit, DFHD2EX1, and GTF trace from the CICS DB2 subtask program DFHD2EX3. All other standard tracing from the attachment is controlled using File Control tracing. A large proportion of the possible trace issued from the CICS attachment facility is exception tracing which is output irrespective of

RI or FC tracing being active, but is issued independently. The levels of FC and RI trace required can be set using the CICS-supplied transaction, CETR, or using SIT parameters STNTRFC= and STNTRRI=.

Figure 33 shows in two parts an example of the trace output from the RMI and the CICS DB2 task related user exit, DFHD2EX1, when level 1 and level 2 RI tracing is active. The trace shows one SQL statement being executed.

CICS DB2 trace output is written to the CICS internal trace table and auxiliary trace and GTF trace destinations if active.

```
AP 2520 ERM ENTRY APPLICATION-CALL-TO-TRUE(DSNCSQL )

               TASK-00042 KE_NUM-0042 TCB-QR   /008BDE88 RET-91002F86 TIME-16:31:52.6697351879 INTERVAL-**.**********        =000001=
                 1-0000  01                                                                    *.                              *
                 2-0000  C4E2D5C3 E2D8D340                                                     *DSNCSQL                        *
                 3-0000  B11E4C99 A62E1E02                                                     *..<rw...                       *

AP 2522 ERM EVENT PASSING-CONTROL-TO-QR-TRUE(DSNCSQL )

               TASK-00042 KE_NUM-0042 TCB-QR   /008BDE88 RET-91002F86 TIME-16:31:52.6697480629 INTERVAL-00.0000128750        =000002=
                 1-0000  01                                                                    *.                              *
                 2-0000  C4E2D5C3 E2D8D340                                                     *DSNCSQL                        *
                 3-0000  B11E4C99 A62E1E02                                                     *..<rw...                       *
                 4-0000  108C636C 0005F004 10B02204 00000000  008A7000 008A7000 10A0005C 10A0C230  *...%..0.....................*..B.*
                   0020  10B021C0 10B02210 10B02206 10A000D0  10B02200 108C61C0 10B021C8 10B021B0  *...{...........}....../{...H....*
                   0040  10B021AC 10B021EC 108C630B 10B021FD  108C630C 10B021B4 10B021B2 00020000  *..............................*
                 5-0000  D8D9                                                                  *QR                             *

AP 3180 D2EX1 ENTRY - APPLICATION REQUEST - EXEC SQL FETCH

               TASK-00042 KE_NUM-0042 TCB-QR   /008BDE88 RET-8009C81E TIME-16:31:52.6697601879 INTERVAL-00.0000121250        =000003=
                 1-0000  020004                                                                *...                            *
                 2-0000  00DE6EC4 C6C8C4F2 D3D6E340 40404040  E7D7F0F5 1028F680 109BEBE8 1091D030  *..>DFHD2LOT     XP05..6....Y.j}.*
                   0020  00000000 0005EFF0 00000000 109BEC78  10A0C340 00000000 00000000 FF732120  *.......0..........C ............*
                   0040  00000000 00000000 00000000 00000000  00000000 E3C5E2E3 D7F0F540 000C010C  *....................TESTP05 ....*
                   0060  A000C000 00000000 00000000 00000000  C9E8D2F2 E9F2C7F1 B11E4C99 A62E1E02  *..{.............IYK2Z2G1..<rw...*
                   0080  D1E3C9D3 D3C9F140 40404040 40404040  00000000 00000000 C7C2C9C2 D4C9E8C1  *JTILLI1         ........GBIBMIYA*
                   00A0  C9E8C1F2 E3F5C6F8 1E4C99A6 2E1EC6D9  C2400003 000110A0 C2C80000 00000000  *IYA2T5F8.<rw..FRB ......BH......*
                   00C0  00000000 00000000 27050001 04000000  00000000 00000000 00000000 0000      *............................  *
```

Figure 33. Sample trace output from the RMI and the CICS DB2 TRUE (Part 1 of 6)

```
AP 3182 D2EX1 EVENT - POST D2EX3 SUBTASK  FOR DB2ENTRY XP05     USING PLAN TESTP05

               TASK-00042 KE_NUM-0042 TCB-QR   /008BDE88 RET-8009C81E TIME-16:31:52.6697704379 INTERVAL-00.0000102500        =000004=
                 1-0000  010C010C A000C000 00000000                                          *......{.....                    *
                 2-0000  E3C5E2E3 D7F0F540                                                  *TESTP05                         *
                 3-0000  00DE6EC4 C6C8C4F2 D3D6E340 40404040  E7D7F0F5 1028F680 109BEBE8 1091D030  *..>DFHD2LOT     XP05..6....Y.j}.*
                   0020  00000000 0005EFF0 00000000 109BEC78  10A0C340 00000000 00000000 FF732120  *.......0.........C .............*
                   0040  00000000 00000000 00000000 00000000  00000000 E3C5E2E3 D7F0F540 010C010C  *....................TESTP05 ....*
                   0060  A000C000 00000000 00000000 00000000  C9E8D2F2 E9F2C7F1 B11E4C99 A62E1E02  *..{.............IYK2Z2G1..<rw...*
                   0080  D1E3C9D3 D3C9F140 40404040 40404040  00000000 00000000 C7C2C9C2 D4C9E8C1  *JTILLI1         ........GBIBMIYA*
                   00A0  C9E8C1F2 E3F5C6F8 1E4C99A6 2E1EC6D9  C2400003 000110A0 C2C80000 00000000  *IYA2T5F8.<rw..FRB ......BH......*
                   00C0  00000000 00000000 27050001 04000000  00000000 00000000 00000000 0000      *............................... *
                 4-0000  00C86EC4 C6C8C4F2 C5D5E340 40404040  E7D7F0F5 40404040 B11E493D 9E60A701  *.H>DFHD2ENT     XP05    .....-x.*
                   0020  E3C5E2E3 D7F0F540 00000000 00000000  00000000 00000000 D1E3C9D3 D3C9F140  *TESTP05 ................JTILLI1 *
                   0040  00808080 40000000 60FD80EE 4DA95641  B11E493D 9E60A701 00000004 00000003  *.... ...-...(z.......-x........*
                   0060  00000001 00000001 00000000 00000000  00000001 00000001 00000000 00000000  *................................*
                   0080  00000000 00000002 00000001 00000000  00000000 00000000 00000000 00000000  *................................*
                   00A0  00000000 00000000 00000000 1091D030  00000000 00000000 10B02210 00000000  *.............j}.................*
                   00C0  00000000 00000000                                                    *........                        *
                 5-0000  02906EC4 C6C8C4F2 C3E2C240 40404040  B11E4C9E CBBEA701 109575C0 109BEBE8  *..>DFHD2CSB     ..<...x..n.{...Y*
                   0020  10B02210 008A2E88 808A2E00 00000000  109BEC60 00000000 00000000 00000000  *.......h...........-............*
                   0040  00000000 00000000 00000000 00000000  00000000 00000000 00000000 E3C5E2E3  *............................TEST*
                   0060  D7F0F540 D1E3C9D3 D3C9F140 40404040  40404040 C5D5E3D9 E7D7F0F5 F0F0F0F1  *P05 JTILLI1         ENTRXP050001*
                   0080  00000000 B11E4C9E DB7E9703 C7C2C9C2  D4C9E8C1 C9E8C1F2 E3F5C6F8 1E4C99A6  *......<..=p.GBIBMIYAIYA2T5F8.<rw*
                   00A0  2E1E0020 443C0001 00000000 00000000  00000000 00000000 40404040 40404040  *.....................           *
                   00C0  40404040 40404040 FF732120 C6D9C240  00030001 10A0C2C8 00000000 00000000  *        ....FRB ......BH........*
                   00E0  00000000 00002705 00010400 00000000  00000000 00000000 00000000 00000000  *................................*
                   0100  000087E0 00019C40 9015C344 80009120  1091D0FC 1091D174 10A0C2C8 1015CEA8  *..g\... ..C...j..j}..jJ...BH...y*
                   0120  109575D8 1091D0FC 10B02210 109575C0  109BEBE8 1091D030 9015BBE8 1015CBE8  *.n.Q.j}......n.{...Y.j}....Y...Y*
                   0140  9015C11E 9015BF84 00000000 C1D7C940  00000004 003400EF 1091D18C 4C6E0034  *..A....d....API .........jJ.<>..*
                   0160  00103220 4BC4C2F2 00000000 0028C4F0  F0F0F174 008A2E88 1015CEA2 B11E4C9E  *.....DB2......D0001....h...s..<.*
                   0180  DBA2EE03 00040000 042C0004 10B02244  00000000 40404040 40404040 40404040  *.s..................           *
                   01A0  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                                *
                   01C0  40404040 40404040 00000004 1091D240  6E6EE399 81838540 E2A38199 A3406E6E  *        .....jK >>Trace Start >>*
                   01E0  00000001 C9C4C5D5 00000000 00000000  00000002 E2C9C7D5 00000000 00000000  *....IDEN.............SIGN........*
                   0200  00000003 C3E3C8C4 00000000 00000000  00000004 C1D7C940 00000000 00000000  *....CTHD.............API ........*
                   0220  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                   0240  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                   0260  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                   0280  4C4CE399 81838540 C5958440 40404C4C                                    *<<Trace End   <<                *
```

Figure 33. Sample trace output from the RMI and the CICS DB2 TRUE (Part 2 of 6)

```
AP 3183 D2EX1 EVENT - AWOKEN BY D2EX3 SUBTASK FOR DB2ENTRY XP05     USING PLAN TESTP05

               TASK-00042 KE_NUM-0042 TCB-QR   /008BDE88 RET-8009C81E TIME-16:31:52.6707882504 INTERVAL-00.0010178125        =000005=
                 1-0000  010C010C A000C000 00000000 00000000  00000000                        *......{.............            *
                 2-0000  E3C5E2E3 D7F0F540                                                  *TESTP05                         *
                 3-0000  00DE6EC4 C6C8C4F2 D3D6E340 40404040  E7D7F0F5 1028F680 109BEBE8 1091D030  *..>DFHD2LOT     XP05..6....Y.j}.*
                   0020  00000000 0005EFF0 00000000 109BEC78  10A0C340 00000000 00000000 FF732120  *.......0.........C .............*
                   0040  00000000 00000000 00000000 00000000  00000000 E3C5E2E3 D7F0F540 010C010C  *....................TESTP05 ....*
                   0060  A000C000 00000000 00000000 00000000  C9E8D2F2 E9F2C7F1 B11E4C99 A62E1E02  *..{.............IYK2Z2G1..<rw...*
                   0080  D1E3C9D3 D3C9F140 40404040 40404040  00000000 00000000 C7C2C9C2 D4C9E8C1  *JTILLI1         ........GBIBMIYA*
                   00A0  C9E8C1F2 E3F5C6F8 1E4C99A6 2E1EC6D9  C2400003 000110A0 C3400000 00000000  *IYA2T5F8.<rw..FRB ......C ......*
                   00C0  00000000 00000000 27050001 04000000  00000000 00000000 00000000 0000      *............................... *
                 4-0000  00C86EC4 C6C8C4F2 C5D5E340 40404040  E7D7F0F5 40404040 B11E493D 9E60A701  *.H>DFHD2ENT     XP05    .....-x.*
                   0020  E3C5E2E3 D7F0F540 00000000 00000000  00000000 00000000 D1E3C9D3 D3C9F140  *TESTP05 ................JTILLI1 *
                   0040  00808080 40000000 60FD80EE 4DA95641  B11E493D 9E60A701 00000004 00000003  *.... ...-...(z.......-x........*
                   0060  00000001 00000001 00000000 00000000  00000001 00000001 00000000 00000000  *................................*
                   0080  00000000 00000002 00000001 00000000  00000000 00000000 00000000 00000000  *................................*
                   00A0  00000000 00000000 00000000 1091D030  00000000 00000000 10B02210 00000000  *.............j}.................*
                   00C0  00000000 00000000                                                    *........                        *
```

Figure 33. Sample trace output from the RMI and the CICS DB2 TRUE (Part 3 of 6)

```
                5-0000  02906EC4 C6C8C4F2 C3E2C240 40404040  B11E4C9E CBBEA701 109575C0 109BEBE8  *..>DFHD2CSB     ..<...x..n.{...Y*
                  0020  10B02210 008A2E88 808A2E00 00000000  109BEC60 00000000 00000000 00000000  *.......h..........-............*
                  0040  00000000 00000000 00000000 00000000  00000000 00000000 00000000 E3C5E2E3  *............................TEST*
                  0060  D7F0F540 D1E3C9D3 D3C9F140 40404040  40404040 C5D5E3D9 E7D7F0F5 F0F0F0F1  *P05 JTILLI1         ENTRXP050001*
                  0080  00000000 B11E4C9E DB7E9703 C7C2C9C2  D4C9E8C1 C9E8C1F2 E3F5C6F8 1E4C99A6  *......<..=p.GBIBMIYAIYA2T5F8.<rw*
                  00A0  2E1E0020 443C0001 00000000 00000000  00000000 00000000 40404040 40404040  *.....................         *
                  00C0  40404040 40404040 FF732120 C6D9C240  00030001 10A0C340 00000000 00000000  *        ....FRB ......C ........*
                  00E0  00000000 00002705 00010400 00000000  00000000 00000000 00000000 00000000  *................................*
                  0100  000087E0 00019C40 9015C344 80009120  1091D0FC 1091D174 10A0C340 1015CEA8  *..g\... ..C...j..j}..jJ...C ...y*
                  0120  109575D8 1091D0FC 10B02210 109575C0  109BEBE8 1091D030 9015BBE8 1015CBE8  *.n.Q.j}......n.{...Y.j}....Y...Y*
                  0140  9015C11E 9015BF84 00000000 C1D7C940  00000005 003400EF 1091D18C 4C6E0034  *..A....d....API .........jJ.<>..*
                  0160  00103220 4BC4C2F2 00000000 0028C4F0  F0F0F174 008A2E88 1015CEA2 B11E4CD1  *.....DB2......D0001....h...s..<J*
                  0180  3F72E502 00040000 042C0004 10B02244  00000000 40404040 40404040 40404040  *..V.................             *
                  01A0  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                                *
                  01C0  40404040 40404040 00000005 1091D250  6E6EE399 81838540 E2A38199 A3406E6E  *        .....jK&>>Trace Start >>*
                  01E0  00000001 C9C4C5D5 00000000 00000000  00000002 E2C9C7D5 00000000 00000000  *....IDEN.............SIGN........*
                  0200  00000003 C3E3C8C4 00000000 00000000  00000004 C1D7C940 00000000 00000000  *....CTHD.............API ........*
                  0220  00000005 C1D7C940 00000000 00000000  00000000 00000000 00000000 00000000  *....API .........................*
                  0240  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                  0260  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                  0280  4C4CE399 81838540 C5958440 40404C4C                                      *<<Trace End   <<                *
```

Figure 33. Sample trace output from the RMI and the CICS DB2 TRUE (Part 4 of 6)

```
AP 3181 D2EX1 EXIT - APPLICATION REQUEST - EXEC SQL FETCH

               TASK-00042 KE_NUM-0042 TCB-QR   /008BDE88 RET-8009C81E TIME-16:31:52.6707984379 INTERVAL-00.0000101875        =000006=
                 1-0000  020004                                                                  *...                             *
                 2-0000  00DE6EC4 C6C8C4F2 D3D6E340 40404040  E7D7F0F5 1028F680 109BEBE8 1091D030  *..>DFHD2LOT     XP05..6....Y.j}.*
                   0020  00000000 0005EFF0 00000000 109BEC78  10A0C340 00000000 00000000 FF732120  *.......0..........C ............*
                   0040  00000000 00000000 00000000 00000000  00000000 E3C5E2E3 D7F0F540 010C010C  *...................TESTP05 ....*
                   0060  A000C000 00000000 00000000 00000000  C9E8D2F2 E9F2C7F1 B11E4C99 A62E1E02  *..{.............IYK2Z2G1..<rw...*
                   0080  D1E3C9D3 D3C9F140 40404040 40404040  00000000 00000000 C7C2C9C2 D4C9E8C1  *JTILLI1         ........GBIBMIYA*
                   00A0  C9E8C1F2 E3F5C6F8 1E4C99A6 2E1EC6D9  C2400003 000110A0 C3400000 00000000  *IYA2T5F8.<rw..FRB ......C ......*
                   00C0  00000000 00000000 27050001 04000000  00000000 00000000 00000000 0000      *............................  *
                 3-0000  00                                                                       *.                               *
                 4-0000  00C86EC4 C6C8C4F2 C5D5E340 40404040  E7D7F0F5 40404040 B11E493D 9E60A701  *.H>DFHD2ENT     XP05    .....-x.*
                   0020  E3C5E2E3 D7F0F540 00000000 00000000  00000000 00000000 D1E3C9D3 D3C9F140  *TESTP05 .................JTILLI1 *
                   0040  00808080 40000000 60FD80EE 4DA95641  B11E493D 9E60A701 00000004 00000003  *.... ...-...(z.......-x.........*
                   0060  00000001 00000001 00000000 00000000  00000001 00000001 00000000 00000000  *................................*
                   0080  00000000 00000002 00000001 00000000  00000000 00000000 00000000 00000000  *................................*
                   00A0  00000000 00000000 00000000 1091D030  00000000 00000000 10B02210 00000000  *.............j}.................*
                   00C0  00000000 00000000                                                        *........                        *
                 5-0000  02906EC4 C6C8C4F2 C3E2C240 40404040  B11E4C9E CBBEA701 109575C0 109BEBE8  *..>DFHD2CSB     ..<...x..n.{...Y*
                   0020  10B02210 008A2E88 808A2E00 00000000  109BEC60 00000000 00000000 00000000  *.......h..........-............*
                   0040  00000000 00000000 00000000 00000000  00000000 00000000 00000000 E3C5E2E3  *............................TEST*
                   0060  D7F0F540 D1E3C9D3 D3C9F140 40404040  40404040 C5D5E3D9 E7D7F0F5 F0F0F0F1  *P05 JTILLI1         ENTRXP050001*
                   0080  00000000 B11E4C9E DB7E9703 C7C2C9C2  D4C9E8C1 C9E8C1F2 E3F5C6F8 1E4C99A6  *......<..=p.GBIBMIYAIYA2T5F8.<rw*
                   00A0  2E1E0020 443C0001 00000000 00000000  00000000 00000000 40404040 40404040  *.....................         *
                   00C0  40404040 40404040 FF732120 C6D9C240  00030001 10A0C340 00000000 00000000  *        ....FRB ......C ........*
                   00E0  00000000 00002705 00010400 00000000  00000000 00000000 00000000 00000000  *................................*
                   0100  000087E0 00019C40 9015C344 80009120  1091D0FC 1091D174 10A0C340 1015CEA8  *..g\... ..C...j..j}..jJ...C ...y*
                   0120  109575D8 1091D0FC 10B02210 109575C0  109BEBE8 1091D030 9015BBE8 1015CBE8  *.n.Q.j}......n.{...Y.j}....Y...Y*
                   0140  9015C11E 9015BF84 00000000 C1D7C940  00000005 003400EF 1091D18C 4C6E0034  *..A....d....API .........jJ.<>..*
                   0160  00103220 4BC4C2F2 00000000 0028C4F0  F0F0F174 008A2E88 1015CEA2 B11E4CD1  *.....DB2......D0001....h...s..<J*
                   0180  3F72E502 00040000 042C0004 10B02244  00000000 40404040 40404040 40404040  *..V.................             *
                   01A0  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                                *
                   01C0  40404040 40404040 00000005 1091D250  6E6EE399 81838540 E2A38199 A3406E6E  *        .....jK&>>Trace Start >>*
                   01E0  00000001 C9C4C5D5 00000000 00000000  00000002 E2C9C7D5 00000000 00000000  *....IDEN.............SIGN........*
                   0200  00000003 C3E3C8C4 00000000 00000000  00000004 C1D7C940 00000000 00000000  *....CTHD.............API ........*
                   0220  00000005 C1D7C940 00000000 00000000  00000000 00000000 00000000 00000000  *....API .........................*
                   0240  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                   0260  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                   0280  4C4CE399 81838540 C5958440 40404C4C                                      *<<Trace End<<                 *
```

Figure 33. Sample trace output from the RMI and the CICS DB2 TRUE (Part 5 of 6)

```
AP 2523 ERM EVENT REGAINING-CONTROL-FROM-QR-TRUE(DSNCSQL)

               TASK-00042 KE_NUM-0042 TCB-QR   /008BDE88 RET-91002F86 TIME-16:31:52.6710398129 INTERVAL-00.0002413750        =000007=
                 1-0000  01                                                                     *.                             *
                 2-0000  C4E2D5C3 E2D8D340                                                      *DSNCSQL                       *
                 3-0000  B11E4C99 A62E1E02                                                      *..<rw...                      *
                 4-0000  108C636C 0005F004 10B02204 00000000  008A7000 008A7000 10A0005C 10A0C230  *...%..0.....................*..B.*
                   0020  10B021C0 10B02210 10B02206 10A000D0  10B02200 108C6960 10B021C8 10B021B0  *...{...........}.......-...H....*
                   0040  10B021AC 10B021EC 108C630B 10B021FD  108C630C 10B021B4 10B021B2 00020000  *................................*
                 5-0000  D8D9                                                                   *QR                            *

AP 2521 ERM EXIT APPLICATION-CALL-TO-TRUE(DSNCSQL )

               TASK-00042 KE_NUM-0042 TCB-QR   /008BDE88 RET-91002F86 TIME-16:31:52.6710585004 INTERVAL-00.0000186875        =000008=
                 1-0000  01                                                                     *.                             *
                 2-0000  C4E2D5C3 E2D8D340                                                      *DSNCSQL                       *
                 3-0000  B11E4C99 A62E1E02                                                      *..<rw...                      *
```

Figure 33. Sample trace output from the RMI and the CICS DB2 TRUE (Part 6 of 6)

CSUB trace

The CSUB is a control block used by the CICS DB2 subtask that issues the requests to DB2. As the subtask is a TCB owned by the CICS DB2 attachment facility and not known to the CICS dispatcher or kernel, CICS tracing cannot be performed in the subtask program, DFHD2EX3. DFHD2EX3 does issue one GTF trace to record the point in time it posts back the CICS task on completion of the DB2 request. In addition to this, DFHD2EX3 maintains its own small trace table at the end of the CSUB control block to record the requests it makes to DB2, and the responses to those requests.

The CSUB trace table is 160 bytes in length allowing ten entries of 16 bytes to be written. The trace table wraps when all ten entries have been used. The format of each trace entry is shown in Table 12.

Table 12. Example of CSUB trace table entry

Bytes 0-3	Trace request number	Fullword number of the trace entry written. The number is used to find the latest entry written.
Bytes 4-7	Trace request	Four-character representation of the DB2 request issued. Possible values are: ABRT - Abort request API - SQL or IFI request COMM - Commit request THD - Create thread request ERRH - Error handler request IDEN - Identify request PREP - Prepare request PSGN - Partial signon request SIGN - Full signon request SYNC - Single phase request TERM - Terminate thread request *REC - Subtask ESTAI recovery routine entered
Bytes 8-9	reserved	
Bytes 10-11	frb return code	Two-byte frb return code
Bytes 12-15	frb reason code	Four-byte frb reason code

The CSUB control block is formatted in a CICS system dump. It is also output if RI (RMI) level 2 trace is active in traces output from the CICS task related user exit DFHD2EX1, plus all exception traces from DFHD2EX1.

In the previous trace example, DATA5 from trace point 3181 is the CSUB and is shown in Figure 34. Looking at the character representation on the right hand side, the trace table is delimited by >>Trace Start >> and <<Trace End <<. In this example you can see that an identify, sign-on, create thread, followed by two API requests have been issued to DB2, and all requests were successful and the FRB return code and reason codes for each request are zeros.

```
5-0000  02906EC4 C6C8C4F2 C3E2C240 40404040  AEFD8812 3264A600 0F51C660 0F5728C8  *..>DFHD2CSB     ..h...w...F-...H*
  0020  0F57E200 008A01F0 808A0168 00000000  0F572940 00000000 00000000 00000000  *..S....0.......... ............*
  0040  00000000 00000000 00000000 00000000  00000000 00000000 00000000 E3C5E2E3  *............................TEST*
  0060  D7F0F540 C3C9C3E2 E4E2C5D9 40404040  40404040 C5D5E3D9 E7D7F0F5 F0F0F0F1  *P05 CICSUSER        ENTRXP050001*
  0080  00000000 AEFD8812 7598C200 C7C2C9C2  D4C9E8C1 C9E8C1F2 E3F5C6F8 FD880CF3  *......h..qB.GBIBMIYAIYA2T5F8.h.3*
  00A0  9FD20020 443C0001 00000000 00000000  00000000 00000000 40404040 40404040  *.K.....................        *
  00C0  40404040 40404040 FF6F3740 C6D9C240  00030001 00202368 00000000 00000000  *        .?. FRB ................*
  00E0  00000000 00002705 00010400 00000000  00000000 00000000 00000000 00000000  *................................*
  0100  00008428 00008398 9C109374 80018770  0F5870FC 0F587174 00202368 1C109EB4  *..d...cq..l...g................*
  0120  0F51C678 0F5870FC 0F57E200 0F51C660  0F5728C8 0F587030 9C108C20 1C109C20  *..F.......S...F-...H............*
  0140  9C10914E 9C108FBC 00000000 C1D7C940  00000005 002C00EF 0F58718C 4C6E002C  *..j+........API .............<>..*
  0160  00103220 3BC4C2F2 00000000 008A01F0  1C109EAE AEFD88A0 1A048401 00040000  *.....DB2.......0......h...d.....*
  0180  037C0004 0F57E234 00000000 40404040  40404040 40404040 40404040 40404040  *.@....S.....                    *
  01A0  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                                *
  01C0  00000005 0F587250 00000000 00000000  6E6EE399 81838540 E2A38199 A3406E6E  *.......&;.......>>Trace Start >>*
  01E0  00000001 C9C4C5D5 00000000 00000000  00000002 E2C9C7D5 00000000 00000000  *....IDEN............SIGN........*
  0200  00000003 C3E3C8C4 00000000 00000000  00000004 C1D7C940 00000000 00000000  *....CTHD............API .........*
  0220  00000005 C1D7C940 00000000 00000000  00000000 00000000 00000000 00000000  *....API .......................*
  0240  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
  0260  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
  0280  4C4CE399 81838540 C5958440 40404C4C                                       *<<Trace End   <<                *
```

Figure 34. Sample CSUB trace

CSUB abend information

If a CICS-DB2 subtask abends, part of the recovery process is to save information from the MVS SDWA in the CSUB control block, for example CSB_SDWA_REGS (regs 0 - 15) and CSB_SDWA_PSW. Although the CSUB is typically freemained following the failure, the exception trace written at the time of failure (AP 319D) captures the CSUB control block containing the SDWA information.

Dump

Control blocks from the CICS DB2 attachment facility are formatted out in a CICS system dump under the control of the DB2 keyword of the CICS IPCS verbexit. The DB2 keyword may specify the following values:

- 0 - no DB2 output
- 1 - summary information only
- 2 - control blocks only
- 3 - summary and control blocks

In a CICS transaction dump, no summary or control blocks appear but the trace table contains the CICS DB2 trace entries.

A sample showing CICS DB2 summary information from a CICS system dump is shown in Figure 35 on page 130.

```
===DB2: CICS/DB2 - SUMMARY
==DB2: GLOBAL STATE SUMMARY
   Db2conn name:                          RCTJT
   Connection status:                     Connected
   In standby mode:                       No
   DB2 id:                                DC2D
   DB2 release:                           0410
   Service task started:                  Yes
   Master subtask started:                Yes
   Tcb limit:                             112
   Currently active tcbs:                 2
   Message Queue1:                        CDB2
   Message Queue2:
   Message Queue3:
   Statistics Queue:                      CDB2
   Standbymode:                           Reconnect
   Connecterror:                          Sqlcode
==DB2: TRANSACTION SUMMARY
Tran Task  TcaAddr  TieAddr  LotAddr  Rctename RcteAddr CsubAddr Correlation  Uowid            Subtask Subtask
id   num                                                  id                               running in DB2
-------------------------------------------------------------------------------------------------------------
XC06 00049 00052080 0F5862E0 0F586358 XC06     0F572800 0F5872C0 ENTRXC060002 AEFD9A4879CF4005 No      No
XP05 00046 00050080 0F586188 0F586200 XP05     0F5728C8 0F587030 ENTRXP050001 AEFD9A408DBE8803 No      No
```

Figure 35. Dump example

DB2 thread identification

A thread executing in DB2 on behalf of a CICS transaction is identified by its *correlation ID* set by the CICS DB2 Attachment Facility. DB2 allows up to 12 bytes to be used for the correlation ID. In releases of CICS before CICS Transaction Server Release 1 Version 2, only 8 bytes were used.

The format 12 byte correlation ID is made up as as *eeeetttnnnn* where *eeee* is either COMD, POOL or ENTR indicating whether it is a command, pool or DB2ENTRY thread; *tttt* is the transid, and *nnnn* is a unique number.

Note: A correlation ID passed to DB2 can be changed only by the CICS Attachment Facility issuing a signon to DB2. If signon reuse occurs by a thread using a primary authorization ID which remains constant across multiple transactions (for example, by using AUTHID(name)) only one signon will occur. In this instance the *tttt* in the correlation ID does not match the running transaction ID. It is the ID of the transaction for which the initial signon occurred.

Transaction abend codes

In releases of CICS before CICS Transaction Server for OS/390 Release 2, the CICS DB2 attachment facility used a single transaction abend code DSNC for various error situations relying on state left in control blocks to distinguish between the error cases. The CICS DB2 attachment facility now uses multiple abend codes. Each abend code is unique to a particular error. The transaction abend codes are AD2x or AD3x and are documented in the *CICS Messages and Codes* manual and are available using the CICS-supplied transaction, CMAC.

Execution Diagnostic Facility (EDF)

The CICS DB2 task related user exit, DFHD2EX1, is enabled with the FORMATEDF keyword and is called by CICS to format the screen for SQL API requests when the transaction is being run under EDF.

In EDF mode, the CICS attachment facility:

- Stops on every EXEC SQL command and deciphers the SQL statement showing it in a file on the panel
- Shows the results before and after SQL calls are processed
- Displays the following:
  - The type of SQL statement.
  - Any input and output variables.
  - The contents of the SQLCA.
  - Primary and secondary authorization IDs. (This helps diagnose SQLCODE -922.)

An EDF panel displays a maximum of 55 variables, which is about ten screens. Each EDF SQL session requires 12KB of CICS temporary storage, which is freed on exit from EDF.

EDF screens for SQL statements are shown in Figure 36, and Figure 37 on page 132.

```
TRANSACTION: XC05 PROGRAM: TESTC05 TASK:0000097 APPLID: CICS41 DISPLAY:00
 STATUS:  ABOUT TO EXECUTE COMMAND

 EXEC SQL OPEN
  DBRM=TESTC05,  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 36. EDF example of the "before" SQL EXEC panel

```
TRANSACTION: XC05 PROGRAM: TESTC05 TASK:0000097 APPLID: CICS41 DISPLAY:00
 STATUS:  COMMAND EXECUTION COMPLETE
 CALL TO RESOURCE MANAGER DSNCSQL
 EXEC SQL OPEN                               P.AUTH=SYSADM  , S.AUTH=
 PLAN=TESTC05,  DBRM=TESTC05,  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 37. EDF example of the "after" SQL EXEC panel

Chapter 12. Monitoring, tuning, and handling deadlocks

Performance considerations in a CICS DB2 environment are described in the following:

This chapter contains Diagnosis, Modification or Tuning information.

"Chapter 10. Application design considerations" on page 95, also discusses some topics which have performance implications.

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 DB2ENTRYs and the pool.
- The level of thread reuse for protected threads in DB2ENTRYs.

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.

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
  - CICS DB2 statistics
- 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)™.

The following sections provide a description of these facilities and the data they provide.

Monitoring the CICS attachment facility

You monitor the CICS attachment facility by using commands addressed to both DB2 and the CICS attachment facility itself.

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. For more information, see “CICS DB2 transaction DSNC” on page 21, and in particular, refer to the DSNC DISPLAY and DSNC MODIFY commands.

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*

For example, the DSNC -DIS THREAD command can shows CICS DB2 threads.

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) of a hyphen, (-), 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.

For more information see “CICS DB2 transaction DSNC” on page 21.

Monitoring the CICS transactions

You can use the CICS auxiliary trace facility to trace SQL calls issued by a CICS application program.

For more information about trace output by the CICS DB2 attachment facility, see “Chapter 11. Problem determination” on page 121.

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 (DB2PM) program product is useful to provide reports based on:

Statistics records
Accounting records
Performance records

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.

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. For more information about activating these facilities and directing the output to SMF and GTF, see “Issuing commands to DB2 using DSNC” on page 21, and “Starting GTF for DB2 accounting, statistics and tuning” on page 20.

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 the *DB2 for OS/390: 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.

```
                    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 38. Sample statistics summary report (Part 1 of 2)

```
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 38. Sample statistics summary report (Part 2 of 2)

The *DB2 for OS/390: 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 38 on page 136 shows a sample of the report provided by DB2PM. Refer to the *DB2 for OS/390: Administration Guide* for additional information on these reports.

Figure 38 on page 136 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)

For performance, thread reuse is recommended 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 more information about deadlocks, see “Handling deadlocks” on page 151.

This statistical data is not checkpointed and is not retained by DB2 across restarts.

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.

The procedures involved in collecting this data are described in “Starting SMF for DB2 accounting, statistics and tuning” on page 19, and in “Starting GTF for DB2 accounting, statistics and tuning” on page 20.

Refer to the *DB2 for OS/390: Administration Guide* for details on the general structure of DB2 SMF and GTF records, and descriptions of the fields in the accounting record.

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. “Accounting considerations for DB2” on page 86 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 39 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, evidence 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 9. Accounting” on page 79 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. In the example, 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.

```
                    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 39. Accounting detail report for a CICS transaction (Part 1 of 2)

```
                                          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 39. Accounting detail report for a CICS transaction (Part 2 of 2)

Package accounting and performance

Accounting information for packages is available from DB2 3.1 onwards, together with enhanced performance analysis tools with support for packages.

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 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:

- 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 41 on page 144).
- 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 42 on page 145) and MAINPACK (see Figure 40 on page 143).
- 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 41 on page 144 shows a sample of an accounting short trace where two packages are involved. The plan name is PLANAPP1.

Figure 42 on page 145 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.

Figure 43 on page 145 shows the accounting short report ordered by plan name. Because the plan name is PLANAPP1, data for different packages is summarized under this plan.

Figure 40 on page 143 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 43 on page 145.

```
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 40. DB2PM accounting short report ordered by MAINPACK within plan name

```
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 41. DB2PM accounting short trace

```
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 42. DB2PM accounting short report ordered by PACKAGE

```
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 43. DB2PM accounting short report ordered by plan name

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.

Monitoring thread reuse

From a performance viewpoint it is important to be able to reuse a thread to avoid the overhead of creating a thread for each CICS transaction or each unit of work. Thread reuse is reported at a DB2ENTRY level, and for the pool in CICS DB2 statistics.

Monitoring CICS

CICS provides monitoring and accounting facilities for the resources needed by CICS transactions within the CICS address space. Two types of records can be produced:

- Performance records that record the resources used by each transaction in the CICS address space
- Exception records that record shortages of resources

The CICS performance class monitoring records include the following DB2–related data fields:

- The total number of DB2 EXEC SQL and instrumentation facility interface (IFI) requests issued by a transaction.
- The elapsed time the transaction waited for a DB2 thread to become available.
- The elapsed time the transaction waited for a CICS DB2 subtask to become available.
- The elapsed time the transaction waited for DB2 to service the DB2 requests issued by the transaction.

CICS monitoring is used in the CICS DB2 environment with the DB2 accounting facility, to monitor performance and to collect accounting information.

CICS DB2 statistics

In addition to the limited statistics output by the DSNC DISP STAT command and those output to the STATSQUEUE destination of the DB2CONN during attachment facility shutdown, a more comprehensive set of CICS DB2 statistics can be collected using standard CICS statistics interfaces:

- The EXEC CICS COLLECT statistics command accepts the DB2CONN keyword to allow CICS DB2 global statistics to be collected. CICS DB2 global statistics are mapped by the DFHD2GDS DSECT.
- The EXEC CICS COLLECT statistics command accepts the DB2ENTRY() keyword to allow CICS DB2 resource statistics to be collected for a particular DB2ENTRY. CICS DB2 resource statistics are mapped by the DFHD2RDS DSECT.
- The EXEC CICS PERFORM STATISTICS command accepts the DB2 keyword to allow the user to request that CICS DB2 global and resource statistics to be written out to SMF.

The CICS DB2 global and resource statistics are described in detail in the *CICS Performance Guide*.

CICS DB2 statistics are supported for all types of CICS statistics namely:

- Requested statistics - CICS DB2 statistics are written as a result of an EXEC CICS PERFORM STATISTICS RECORD command with the DB2 keyword.
- Requested reset statistics - a special case of requested statistics in which the statistics counters are reset after collection.
- Interval statistics - statistics written when a requested interval expires.

- End of day statistics - a special case of interval statistics.
- Unsolicited statistics - CICS writes DB2 global and resource statistics to SMF when the attachment facility is shut down. Also, DB2 resource statistics are written to SMF when a DB2ENTRY is discarded.

The CICS sample statistics program, DFH0STAT, supports DB2 statistics. It uses EXEC CICS COLLECT STATISTICS commands with the DB2CONN and DB2ENTRY keywords to collect statistics. It also uses EXEC CICS INQUIRE commands for the DB2CONN and DB2ENTRYs to collect data. An example of the output from DFH0STAT is shown in two parts in Figure 44 on page 148.

```
Applid IYK4Z2G1  Sysid JOHN  Jobname CI13JTD5    Date 11/17/1998  Time 16:21:46                CICS 5.3.0                PAGE     2

DB2 Connection

  DB2 Connection Name. . . . . . . . . . . :  RCTJT
  DB2 Sysid. . . . . . . . . . . . . . . . :  DC2D
  DB2 Release. . . . . . . . . . . . . . . :   4.1.0
  DB2 Connection Status. . . . . . . . . . :  CONNECTED           DB2 Connect Date and Time . . . :  11/17/1998  16:18:50.06058
  DB2 Connection Error . . . . . . . . . . :  SQLCODE
  DB2 Standby Mode . . . . . . . . . . . . :  RECONNECT
  DB2 Pool Thread Plan Name. . . . . . . . :
  DB2 Pool Thread Dynamic Plan Exit Name . :  DSNCUEXT
  Pool Thread Authtype . . . . . . . . . . :  TXID
  Pool Thread Authid . . . . . . . . . . . :
  Command Thread Authtype. . . . . . . . . :  SIGNID
  Command Thread Authid. . . . . . . . . . :
  Signid for Pool/Entry/Command Threads. . :  JTILLI1
  Create Thread Error. . . . . . . . . . . :  N906                Message TD Queue 1. . . . . . . . . . . :  CDB2
  Protected Thread Purge Cycle . . . . . . :  00.30               Message TD Queue 2. . . . . . . . . . . :
  Deadlock Resolution. . . . . . . . . . . :  ROLLBACK           Message TD Queue 3. . . . . . . . . . . :
  Non-Terminal Intermediate Syncpoint. . . :  NORELEASE
  Pool Thread Wait Setting . . . . . . . . :  WAIT                Statistics TD Queue . . . . . . . . . . :  CDB2
  Pool Thread Priority . . . . . . . . . . :  HIGH                DB2 Accounting records by . . . . . . . :  TXID
  Current TCB Limit. . . . . . . . . . . . :         80
  Current number of TCBs . . . . . . . . . :         23
  Peak number of TCBs. . . . . . . . . . . :         23
  Current number of free TCBs. . . . . . . :          2
  Current number of tasks on TCB Readyq. . :          0
  Peak number of tasks on TCB Readyq . . . :          0
  Pool Thread Limit. . . . . . . . . . . . :         50           Number of Calls using Pool Threads. . . . . :          0
  Current number of Pool Threads . . . . . :          0           Number of Pool Thread Signons . . . . . . . :          0
  Peak number of Pool Threads. . . . . . . :          2           Number of Pool Thread Commits . . . . . . . :          0
  Number of Pool Thread Waits. . . . . . . :          0           Number of Pool Thread Aborts. . . . . . . . :          0
                                                                  Number of Pool Thread Single Phase. . . . . :          0
  Current number of Pool Tasks . . . . . . :          0           Number of Pool Thread Reuses. . . . . . . . :          0
  Peak number of Pool Tasks. . . . . . . . :          0           Number of Pool Thread Terminates. . . . . . :          2
  Current Total number of Pool Tasks . . . :          0
  Current number of Tasks on Pool Readyq . :          0
  Peak number of Tasks on Pool Readyq. . . :          0
  Current number of DSNC Command threads . :          0           Number of DSNC Command Calls. . . . . . . . :          0
  Peak number of DSNC Command threads. . . :          0           Number of DSNC Command Signons. . . . . . . :          0
  DSNC Command Thread Limit. . . . . . . . :          2           Number of DSNC Command Thread Terminates. . :          0
                                                                  Number of DSNC Command Thread Overfl ows . .:          0

Applid IYK4Z2G1  Sysid JOHN  Jobname CI13JTD5    Date 11/17/1998  Time 16:21:46                CICS 5.3.0                PAGE     3

DB2 Entries

  DB2Entry Name. . . . . . . . . . . . . . :  XC06                DB2Entry Status . . . . . . . . . . . . :  ENABLED
  DB2Entry Static Plan Name. . . . . . . . :  TESTC06            DB2Entry Disabled Action. . . . . . . . :  POOL
  DB2Entry Dynamic Plan Exit Name. . . . . :                     DB2Entry Deadlock Resolution. . . . . . :  ROLLBACK
  DB2Entry Authtype. . . . . . . . . . . . :  USERID             DB2Entry Accounting records by. . . . . :  UOW
  DB2Entry Authid. . . . . . . . . . . . . :
                                                                 Numberof Calls using DB2Entry. . . . . . . .:      4,000
  DB2Entry Thread Wait Setting . . . . . . :  WAIT               Number of DB2Entry Signons. . . . . . . . . :        500
                                                                 Number of DB2Entry Commits. . . . . . . . . :          0
  DB2Entry Thread Priority . . . . . . . . :  HIGH               Number of DB2Entry Aborts . . . . . . . . . :          0
  DB2Entry Thread Limit. . . . . . . . . . :         20          Number of DB2Entry Single Phase . . . . . . :      1,000
  Current number of DB2Entry Threads . . . :          0          Number of DB2Entry Thread Reuses. . . . . . :        480
  Peak number of DB2Entry Threads. . . . . :         20          Number of DB2Entry Thread Terminates. . . . :          0
                                                                 Number of DB2Entry Thread Waits/Overflows . :        480
  DB2Entry Protected Thread Limit. . . . . . . . . :  20
  Current number of DB2Entry Protected Threads . . :  20
  Peak number of DB2Entry Protected Threads. . . . :  20
  Current number of DB2Entry Tasks . . . . . . . . :   0
  Peak number of DB2Entry Tasks. . . . . . . . . . :  98
  Current Total number of DB2Entry Tasks . . . . . : 500
  Current number of Tasks on DB2Entry Readyq . . . :   0
  Peak number of Tasks on DB2Entry Readyq. . . . . :  78
```

Figure 44. Example output from DFH0STAT (part 1)

```
DB2Entry Name. . . . . . . . . . . . . . :  XP05
DB2Entry Static Plan Name. . . . . . . . :  TESTP05
DB2Entry Dynamic Plan Exit Name. . . . . :
DB2Entry Authtype. . . . . . . . . . . . :  USERID
DB2Entry Authid. . . . . . . . . . . . . :

DB2Entry Thread Wait Setting . . . . . . :  POOL

DB2Entry Thread Priority . . . . . . . . :  HIGH
DB2Entry Thread Limit. . . . . . . . . . :                1
Current number of DB2Entry Threads . . . :                0
Peak number of DB2Entry Threads. . . . . :                1


DB2Entry Protected Thread Limit. . . . . . . . . . :        3
Current number of DB2Entry Protected Threads . . . :        1
Peak number of DB2Entry Protected Threads. . . . . :        1
Current number of DB2Entry Tasks . . . . . . . . . :        0
Peak number of DB2Entry Tasks. . . . . . . . . . . :        3
Current Total number of DB2Entry Tasks . . . . . . :       19
Current number of Tasks on DB2Entry Readyq . . . . :        0
Peak number of Tasks on DB2Entry Readyq. . . . . . :        0
```

```
DB2Entry Status . . . . . . . . . . . . :  ENABLED
DB2Entry Disabled Action. . . . . . . . :  POOL
DB2Entry Deadlock Resolution. . . . . . :  ROLLBACK
DB2Entry Accounting records by. . . . . :  UOW

Number of Calls using DB2Entry. . . . . . . :          57
Number of DB2Entry Signons. . . . . . . . . :          19
Number of DB2Entry Commits. . . . . . . . . :           0
Number of DB2Entry Aborts . . . . . . . . . :          19
Number of DB2Entry Single Phase . . . . . . :           0
Number of DB2Entry Thread Reuses. . . . . . :          16
Number of DB2Entry Thread Terminates. . . . :           0
Number of DB2Entry Thread Waits/Overflows . :           2
```

Figure 45. Example output from DFH0STAT (part 2)

Important statistics fields to look at as regards performance and tuning are:

- The number of calls made using a thread from a DB2ENTRY or the pool and the number of thread reuses, that is, the number of times an existing thread was reused. Thread reuse is good for performance. The use of protected threads can increase thread reuse.
- If THREADWAIT(YES) is specified, the peak number of tasks on the readyq waiting for a thread. It is better to limit transactions using a transaction class rather than allow them to queue for threads.
- On the DB2CONN, check the number of tasks on the pool readyq. Also the peak number of tasks on the TCB readyq. If the latter is nonzero, tasks were queued waiting for a TCB rather than a thread, and the peak number of TCBs equals the TCBLIMIT. It means the sum of the THREADLIMIT values for the pool, for command threads and all DB2ENTRYs, exceeds the number of TCBs allowed. TCBLIMIT or the THREADLIMITs should be adjusted in this case.

Monitoring the overall MVS system

RMF usage is not discussed in this section. It is referred to here as a method of obtaining processor times for CICS and 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 9. Accounting” on page 79.

Tuning

For information on tuning DB2 tables and the DB2 subsystem, and for general considerations when tuning a DB2 application, see the *DB2 for OS/390 Administration GuideDB2 for OS/390: Administration Guide*.

Tuning a CICS application

Tuning a CICS application must be done in two phases:

1. Before moving an application to production
2. 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 are 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 DB2ENTRY definitions required, 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.

Tuning the CICS attachment facility

When tuning the CICS attachment facility, you must understand the underlying architecture.

- 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 are explained in detail in “Chapter 5. Defining the CICS DB2 connection” on page 37.

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, 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. Thread reuse can be measured using CICS DB2 statistics.

  Limit conversational transactions either through transaction classes or by using a dedicated DB2ENTRY (THREADLIMIT greater than 0) with THREADWAIT=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 PRIORITY parameter.
- Choose the best authorization strategy to avoid or minimize the process of sign-on by each thread.
- Minimize the number of DB2ENTRYs. Use wildcarding and dynamic plan selection where relevant to combine appropriate transactions in an entry. Allow low use transactions to default to the pool. However, it should be noted that defining transaction IDs using wilcard characters removes the ability to collect CICS DB2 statistics on a per transaction basis as statistics are collected for each DB2ENTRY which will now represent a group of transactions.

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 DROLLBACK parameter for the DB2CONN (if a transaction is using a pool thread) or the DB2ENTRY used by the transaction. The other partner continues processing after the victim is rolled back.

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.

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.

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 46 on page 153.

```
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 46. Deadlock messages

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.

Finding the resources involved

To find the other resources involved in a deadlock, you may have to activate a DB2 performance trace and recreate the deadlock. Suppose that the reason for solving the deadlock is that the number of deadlocks is too high. Normally recreating 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.

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.

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.

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.

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 47 shows a list of preventive actions and the corresponding main effects. Notes:

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

Figure 47. Deadlock prevention

1. Due to changes in access path.
2. Cursor stability is usually better than repeatable read.

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.

Index

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Z

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To ask questions, make comments about the functions of IBM products or systems, or to request additional publications, contact your IBM representative or your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–842327
  - From within the U.K., use 01962–842327
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Program Number: 5655-147

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-1939-34

Spine information:

CICS TS for OS/390 CICS DB2 Guide Release 3