CICS Transaction Server for z/OS
5.6

Db2 Guide

Note

Before using this information and the product it supports, read the information in Product Legal Notices.

This edition applies to the IBM® CICS® Transaction Server for z/OS®, Version 5 Release 6 (product number 5655-Y305655-BTA) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this PDF

This PDF provides introductory and guidance information on evaluating, installing, and using the CICS Db2® attachment facility, and on defining and maintaining your CICS Db2 environment.

For details of the terms and notation used in this book, see Conventions and terminology used in CICS documentation in IBM Documentation.

Date of this PDF

This PDF was created on 2024-04-22 (Year-Month-Date).

Chapter 1. Overview of the CICS Db2 interface

This section gives an overview of the CICS interface to Db2.

Overview: How CICS connects to Db2

A CICS Db2 attachment facility is provided with CICS. The CICS Db2 attachment facility provides CICS applications with access to Db2 data while operating in the CICS environment.

CICS coordinates recovery of both Db2 and CICS data if transaction or system failure occurs.

The CICS Db2 attachment facility creates an overall connection between CICS and Db2. CICS applications use this connection to issue commands and requests to Db2. 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 can name an individual Db2 subsystem to which CICS connects, or (if you have DB2® Version 7 or later) you can use the group attach facility to let Db2 choose any active member of a data-sharing group of Db2 subsystems for the connection. You also have the option of CICS automatically connecting and reconnecting to Db2. A Db2 system can be shared by several CICS systems, but each CICS system can be connected to only one Db2 subsystem at a time.

You define the CICS Db2 connection using three different CICS resource definitions: DB2CONN (the Db2 connection definition), DB2ENTRY (the Db2 entry definition), and DB2TRAN (the Db2 transaction definition). You must install a DB2CONN resource definition before you can start the CICS Db2 connection. (Do not confuse this resource definition with the DB2CONN system initialization parameter, which specifies whether you want CICS to start the Db2 connection automatically during initialization.) You can also create DB2ENTRY and, if necessary, DB2TRAN definitions to ensure that your important transactions are prioritized. Overview: How you can define the CICS Db2 connection has more information about these resource definitions.

Attachment commands display and control the status of the CICS Db2 attachment facility, and are issued using the CICS supplied transaction DSNC. The attachment commands are:

- STRT - start the connection to Db2
- STOP - stop the connection to Db2
- DISP - display the status of threads, and display statistics
- MODI - modify characteristics of the connection to Db2
- DISC - disconnect threads

The connection between CICS and Db2 is a multithread connection. Within the overall connection between CICS and Db2, there is a thread—an individual connection into Db2—for each active CICS transaction accessing Db2. Threads allow each CICS transaction to access Db2 resources, such as a command processor or an application plan (the information that tells Db2 what the application program's SQL requests are, and the most efficient way to service them). See Overview: How threads work for a full explanation of how threads work.

When an application program operating in the CICS environment issues its first SQL request, CICS and Db2 process the request as follows:

- A language interface, or stub, DSNCLI, that is link-edited with the application program calls the CICS resource manager interface (RMI).
- The RMI processes the request, and passes control to the CICS Db2 attachment facility's task-related user exit (TRUE), the module that invokes Db2 for each task.
- The CICS Db2 attachment facility schedules a thread for the transaction. At this stage, Db2 checks authorization, and locates the correct application plan.
- Db2 takes control, and the CICS Db2 attachment facility waits while Db2 services the request.

- When the SQL request completes, Db2 passes the requested data back to the CICS Db2 attachment facility.
- CICS now regains control, and the CICS Db2 attachment facility passes the data and returns control to the CICS application program.

The Db2 address spaces

Db2 requires several different address spaces.

Figure 1 on page 3 shows these address spaces.

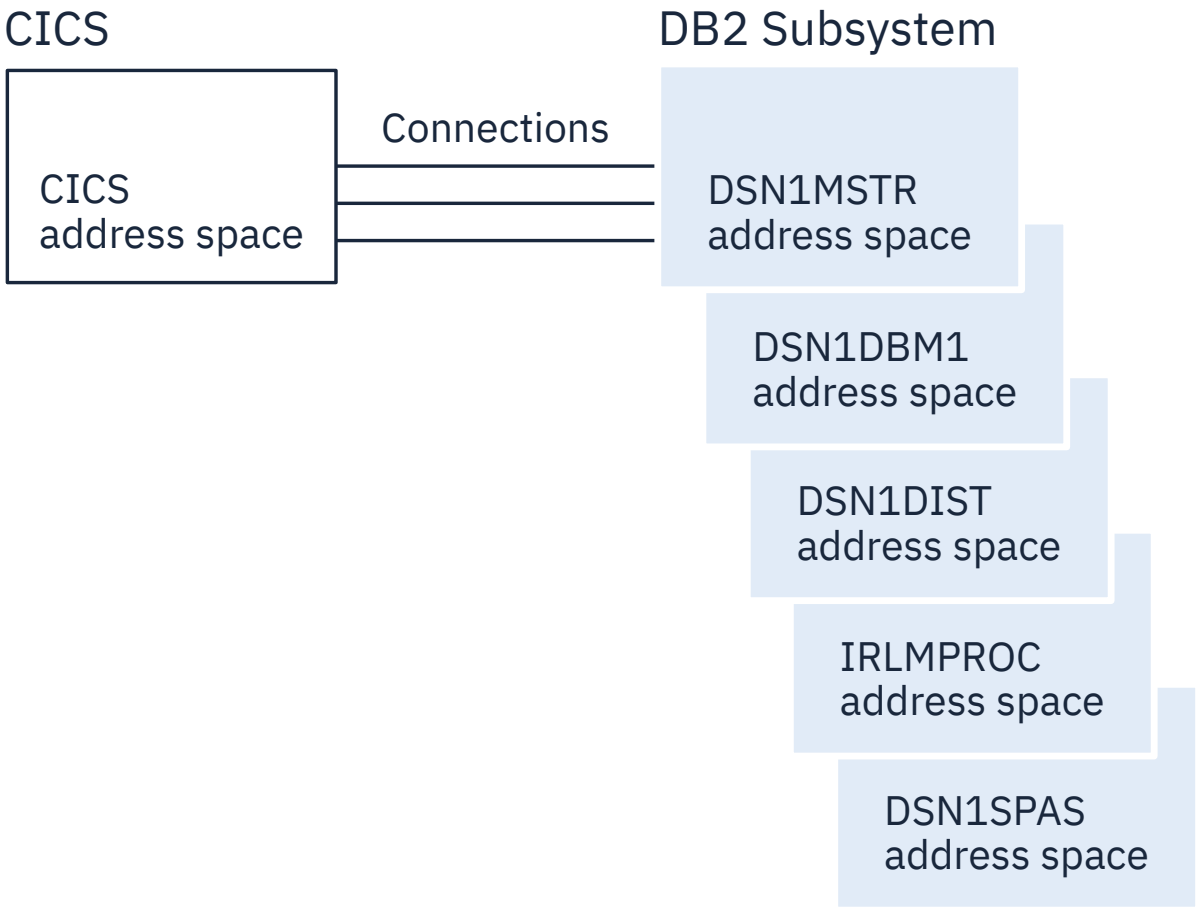


Figure 1. The Db2 address spaces

Various tasks are performed in the different address spaces, as follows:

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.

Overview: How threads work

Each CICS transaction needs its own thread to access Db2.

Threads are created when they are needed by transactions, at the point when the application issues its first SQL or command request. The transaction uses the thread to access resources managed by Db2. When a thread is no longer needed by the transaction, because the transaction has accessed all the resources it needs to use, the thread is released (typically after syncpoint completion). It takes processor resources to create a thread, so when a thread is released, the CICS Db2 attachment facility checks to see if another transaction needs a thread. If another transaction is waiting for a thread, the CICS Db2 attachment facility reuses the existing thread for that transaction to access Db2. If the thread is no longer needed by any transaction, it is terminated, unless you have requested that it should be protected (kept) for a period of time. A protected thread is reused if another transaction requests it within that period of time; if not, it is terminated when the protection time expires.

There are different types of thread, and you can set a limit on the number of each type of thread that can be active at any one time. This prevents the overall CICS Db2 connection from becoming overloaded with work. A special type of thread is used for Db2 commands issued using the DSNC transaction, and you can also define special threads for CICS transactions with particular requirements, such as transactions that require a fast response time. You can define what a transaction must do if no more threads of the type it needs are available — it can wait until a thread of the right type is available; it can use a general-purpose thread, called a pool thread; or it can abend.

The types of thread provided by the CICS Db2 attachment facility are:

Command threads
Command threads are reserved by the CICS Db2 attachment facility for issuing commands to Db2 using the DSNC transaction. They are not used for commands acting on the CICS Db2 attachment facility itself, because these commands are not passed to Db2. When a command thread is not available, commands automatically overflow to the pool, and use a pool thread. Command threads are defined in the command threads section of the DB2CONN definition.

Entry threads
Entry threads are specially defined threads intended for transactions with special requirements, such as transactions that require a fast response time, or transactions with special accounting needs. You can instruct the CICS Db2 attachment facility to give entry threads to particular CICS transactions. You define the different types of entry threads that are needed for different transactions, and you can set a limit on the number of each of these types of entry thread. If a transaction is permitted to use an entry thread, but no suitable entry thread is available, the transaction can overflow to the pool and use a pool thread, or wait for a suitable entry thread, or abend, as you have chosen in the definition for the entry thread.

A certain number of each type of entry thread can be protected. When an entry thread is released, if it is protected it is not terminated immediately. It is kept for a period of time, and if another CICS transaction needs the same type of entry thread during that period, it is reused. This avoids the overhead involved in creating and terminating the thread for each transaction. An entry thread that is

unprotected is terminated immediately, unless a CICS transaction is waiting to use it the moment it is released.

Entry threads are defined using a DB2ENTRY definition.

Pool threads
Pool threads are used for all transactions and commands that are not using an entry thread or a Db2 command thread. Pool threads are intended for low volume transactions, and for overflow transactions that could not obtain an entry thread or a Db2 command thread. 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 definition.

For more detailed information on how the different types of thread are created, used and terminated, see How threads are created, used, and terminated.

Each thread runs under a thread task control block (thread TCB) that belongs to CICS. CICS and Db2 both have connection control blocks linked to the thread TCB. They use these connection control blocks to manage the thread into Db2, and to communicate information to each other about the thread. The Db2 connection control block controls the thread within Db2. The CICS connection control block, called the CSUB, acts as a pointer to the Db2 connection control block, and contains the information CICS requires to call the Db2 connection control block when the thread is needed. Db2 calls these connection control blocks “agent structures”.

While CICS is connecting to a Db2 subsystem, the CICS Db2 task-related user exit (the module that invokes Db2 for each task) is automatically enabled as open API, so it can use the open transaction environment (OTE).

Both types of TCB that the CICS Db2 attachment facility uses to run the threads, the open TCBs and the subtask TCBs, are referred to in this documentation as “thread TCBs”. In many situations, the different nature of the two types of thread TCB does not lead to any differences in the operation of the CICS Db2 connection. Where the different types of thread TCB do cause the CICS Db2 connection to behave differently, a distinction is made between the two types. For more technical information on thread TCBs, see Thread TCBs (task control blocks).

Thread TCBs in the open transaction environment

When CICS is connected to DB2 Version 7 or later, or JDBC 2.0 or later, it uses the open transaction environment. In this environment, the CICS Db2 attachment facility uses open TCBs (L8 mode) as the thread TCBs, rather than using specially created subtask TCBs.

Open TCBs perform other tasks besides accessing Db2 resources. In the open transaction environment, the CICS Db2 task-related user exit runs on an open TCB rather than on the CICS main TCB. If the application program that made the Db2 request is threadsafe, it can also run on the open TCB. (See Enabling CICS Db2 applications to use OTE through threadsafe programming for more information on application programs in the open transaction environment.)

Open TCBs are not permanently associated with a CSUB and Db2 connection control block. The CICS Db2 attachment facility can associate them with any CSUB and Db2 connection control block that are available, and therefore with any thread that is available. When the open TCB no longer needs the connection to Db2, it dissociates from, or releases, the thread, CSUB and Db2 connection control block. The thread, CSUB and Db2 connection control block can then be used by a different open TCB for the tasks it wants to perform in Db2.

If the thread is terminated before it is reused, the CSUB and Db2 connection control block remain available in the system. If no existing threads are available, and an open TCB needs a connection to Db2, the CICS Db2 attachment facility can associate the unused CSUB and Db2 connection control block with the open TCB, and reuse them to run a new thread into Db2.

Figure 2 on page 6 summarizes how thread TCBs operate in the open transaction environment.

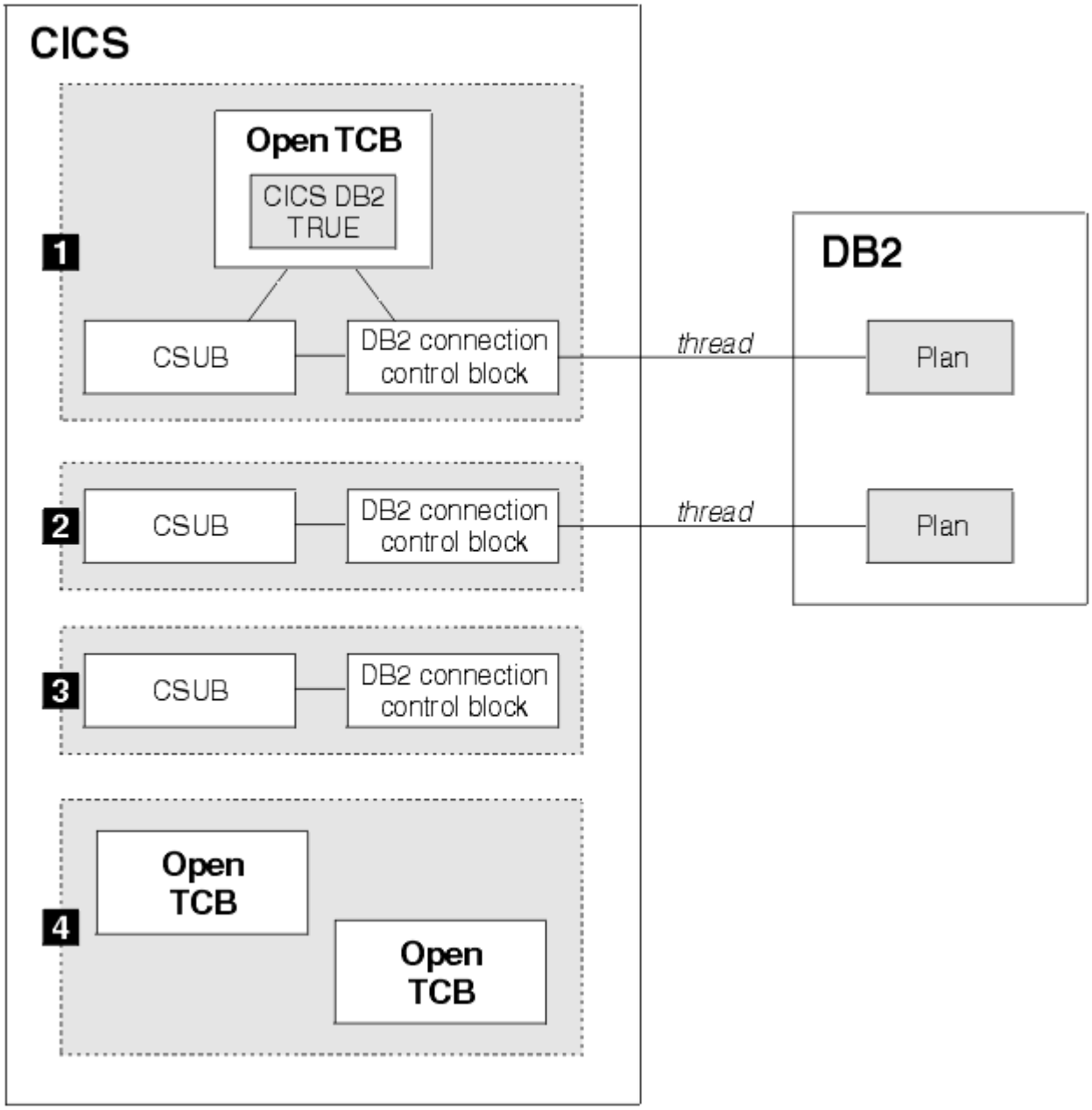


Figure 2. Thread TCBs in the open transaction environment

In Figure 2 on page 6, situation 1 shows CICS using a thread to access Db2 in the open transaction environment. The CICS Db2 task-related user exit has been invoked by the Resource Manager Interface (RMI), and it is operating on an open TCB. The CICS Db2 attachment facility has associated a CSUB and a Db2 connection control block with the open TCB. The Db2 connection control block has a thread into Db2. The plan associated with the thread is held in Db2.

Situation 2 shows a thread that is not currently in use, but is protected. The CSUB and Db2 connection control block are still linked to each other and have a thread, but no open TCB is attached to them. The thread is available for reuse.

Situation 3 shows an assembly that remains after a thread was terminated. The CSUB and Db2 connection control block are available for reuse. They need a new thread.

Situation 4 shows open TCBs that are available for reuse. The CICS Db2 attachment facility can use these open TCBs and associate CSUB and Db2 connection control block assemblies with them to run threads into Db2.

Overview: Enabling CICS application programs to access Db2

A CICS application program that accesses Db2 must be prepared in the same way as a normal CICS application program, and must also go through the Db2 bind process to produce an application plan.

About this task

The Db2 bind process produces an application plan (often just called a “plan”). Each plan contains the bound form of all the SQL statements from the application programs that use it, and it allows those application programs to access Db2 data at run time.

The plans are held in Db2, and each thread into Db2 relates to a plan. The plan that each type of thread uses is named on the DB2CONN definition (for pool threads) or the thread's DB2ENTRY definition (for entry threads). The plan for a particular type of thread must contain the bound form of the SQL statements from all the application programs that use that type of thread to access Db2. You can either name the plan explicitly or name a dynamic plan exit. A dynamic plan exit is a routine that determines which plan to use for the transaction that has requested a thread of that type.

Preparing a CICS application program that accesses Db2

A CICS application program that access DB must be put through a number of processes to build a DBRM and create the application load module before it enters the bind process.

About this task

Figure 3 on page 8 shows the steps in preparing a CICS application program to access Db2.

The first step is to put the program through the Db2 precompiler. The Db2 precompiler builds a database request model (DBRM) that contains information about each of the program's SQL statements.

The second, third and fourth steps are the normal process for preparing any CICS application program, whether or not it accesses Db2. The second step is to put the program through the CICS command language translator. The third step is to compile or assemble the program. The fourth step is to link-edit the program with the necessary interfaces (including the CICS Db2 language interface module DSNCLI). The end product of Steps 2, 3 and 4 is an application load module that enables the program to run. For more information on these steps, see CICS Db2 program preparation.

An extra step is required to enable the program to use the information in the DBRM that was created in Step 1. This fifth step is the bind process. The bind process requires Db2, and it uses the DBRM to produce an application plan that enables the program to access Db2 data. See “The bind process” on page 9 for an explanation of the bind process.

If you are using one of the Language Environment®-conforming compilers for COBOL and PL/I, you can combine some of these steps into a single task, because these compilers have integrated the CICS command language translator, and (depending on your version of Db2) an SQL statement coprocessor. See CICS Db2 program preparation for more information.

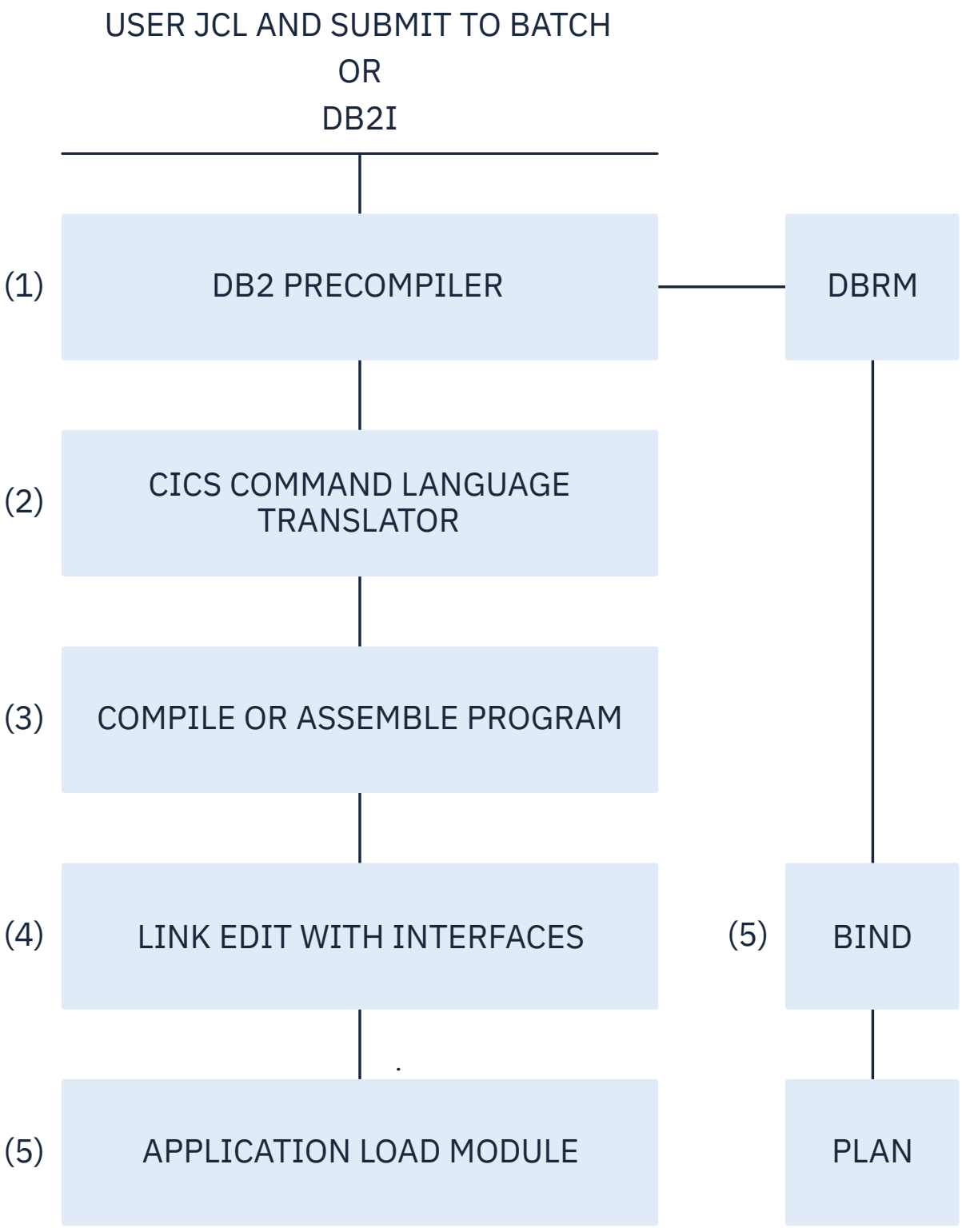


Figure 3. Steps to prepare a CICS application program that accesses Db2

The bind process

Before a CICS application that accesses Db2 can run, it must go through the bind process.

For the bind process, you require:

- Db2
- The DBRM (database request module) produced by the Db2 precompiler for each program in the application.

The DBRM contains the SQL statements that the Db2 precompiler has extracted from the application program. In the bind process, the SQL statements in the DBRM are put into an operational ("bound") form, by being translated into the control structures that Db2 uses when it runs SQL statements. The resulting material can be made into a package, or it can be placed straight into an application plan (see "Plans, packages and dynamic plan exits" on page 9). The whole process is called "binding" the DBRM. For more information on binding, see Programming for Db2 for z/OS in Db2 for z/OS product documentation. See Bind options and considerations for programs for more detail on options you should choose during the bind process in the CICS Db2 environment.

Plans, packages and dynamic plan exits

An application plan allows application programs to access Db2 data at execution time. The plan contains the bound, operational, form of the SQL statements from the DBRMs that were built from the application programs. It also relates the whole application process to the local instance of Db2 where it will be used.

The operational SQL statements from a DBRM can be placed straight into the plan, in which case we say that the DBRM is bound into a plan. Alternatively, you can bind a DBRM into a package (using the BIND PACKAGE command), which contains the operational SQL statements from a single DBRM. You can group related packages into collections. You can then include the package name or collection name in a list of packages, and bind the list of packages into the plan. A single plan can contain both a package list, and DBRMs bound directly into the plan. You can create a plan using the DBRMs from a single CICS application, or you can use the DBRMs from more than one application to create a single plan. Designing the relationship between CICS applications and Db2 plans and packages discusses the advantages and disadvantages of different designs for your plans and packages.

As well as being created, application plans need to be maintained. If the SQL statements change in one or more of the application programs using a plan, you need to rebuild the DBRMs for the changed application programs. If you bound the old versions of those DBRMs directly into your plan, you need to identify all the DBRMs that are bound directly into that plan, for both the changed programs and any unchanged programs, and bind them all into the plan again. While you are binding the DBRMs into the plan, applications cannot use the plan to access Db2. However, if you bound the old versions of the DBRMs for the changed application programs into packages, and then included the names of the packages (or of the collections containing them) on the package list in the plan, you do not need to bind any other packages or directly-bound DBRMs into the plan again. You bind the new versions of the DBRMs for the changed application programs into packages with the same names as the old versions. You do not need to bind the plan again—it locates the new versions of the packages. While you are changing the packages, application programs can still use the other packages and directly-bound DBRMs in the plan. See What to bind after a program change for more information on maintaining plans.

Each thread into Db2 relates to a plan—see "Overview: How threads work" on page 4 for more information about threads. The plan that each type of thread uses is named on the DB2CONN definition (for pool threads) or the thread's DB2ENTRY definition (for entry threads). When CICS requests the use of a thread for an application to access Db2, it tells Db2 the name of the plan associated with that type of thread, and Db2 locates the plan. The definition for each type of thread can either name a specific plan, or it can name a dynamic plan exit, a routine that determines which plan to use for the transaction that has requested the thread.

If the definition of the pool thread or entry thread names a specific plan, all the transactions that use that type of thread must use that plan. The transactions that can use a type of entry thread are specified in the DB2ENTRY and DB2TRAN definitions for the thread. If the DB2ENTRY definition for the thread names a specific plan, the DBRMs from all the application programs that could run under all those transaction

IDs must be bound into the same plan, or bound into packages that are then listed in the same plan. If the DBRMs from any of the application programs that run under those transaction IDs are bound directly into the plan, and you change the SQL statements in any of those application programs, the whole plan will be inaccessible while you bind all the directly-bound DBRMs into the plan again. This means that no transaction can use that type of entry thread while you are maintaining the plan. Pool threads could be used by *any* of your CICS applications that access Db2, so if the DB2CONN definition names a specific plan for the pool threads, the plan needs to be prepared and maintained using the DBRMs from all those applications. If any of the DBRMs have been bound directly into the plan, the whole plan will be inaccessible during maintenance, and no transaction will be able to use a pool thread.

There are two ways to avoid making types of thread unavailable while you are maintaining plans. The best solution is to avoid binding DBRMs directly into plans, by using packages instead. If you bind each separate DBRM as a package and include them in package lists in plans, the plans are still accessible while you are maintaining individual packages. While you are carrying out maintenance work on a particular program, the pool threads or entry threads related to plans involving that program are still available, because the plans are still accessible. This means that you can safely name a specific plan for each thread. If you want to start using packages, see Using Db2 packages and Programming for Db2 for z/OS in Db2 for z/OS product documentation for details of how to implement packages.

An alternative solution, developed before packages were available in Db2, is to use a dynamic plan exit. Using a dynamic plan exit means that you do not have to name a specific plan for each type of thread, so even when a particular plan is inaccessible during maintenance, the threads are still available. To implement this solution, you create many small plans for your CICS applications, each containing the DBRMs from a few closely-related programs. Then, instead of specifying a plan name in the PLAN attribute of the DB2CONN or DB2ENTRY definition for each type of thread, you specify an exit program in the PLANEXITNAME attribute. When an application program issues its first SQL statement and a certain type of thread is requested, the exit program that you have specified in the thread definition selects the plan to use for that application program. If a particular plan is inaccessible during maintenance, the application programs requiring that plan cannot use a thread, but other application programs can use the same type of thread with their own plans. However, note that once a particular instance of a type of thread has been created for an application program to use, that thread instance is associated with the plan that the dynamic plan exit selected. For another application program to reuse the thread, it must use the same plan. If the dynamic plan exit selects a different plan for the application program, it must find or create a different thread with the correct plan. This reduces the opportunities for thread reuse. See Dynamic plan exits for more information on dynamic plan exits.

Chapter 2. Defining the CICS Db2 connection

The structure and performance of your CICS Db2 connection is determined when you define the DB2CONN, DB2ENTRY, and DB2TRAN objects.

Before you begin

For CICS to connect to Db2, you must add one or both Db2 load libraries to the CICS startup job, as needed:

- The *db2hlq*.SDSNLOAD Db2 library, so that the CICS Db2 attachment facility is able to load the Db2 program request handler, DSNAPRH.
- The *db2hlq*.SDSNLOD2 Db2 library, required if you want to use JDBC or SQLJ with a Db2 type 2 connection. This is in addition to the SDSNLOAD library required for non-Java™ programs.

  The Liberty Db2 data source with type 4 connectivity does not use the CICS Db2 connection resource, so there is no requirement for any additional CICS configuration.

Place the Db2 load libraries in the MVS™ link list, or add it to the STEPLIB concatenation of the CICS job. *db2hlq* is the high-level qualifier for Db2 libraries. If you use STEPLIB, add the Db2 library after the CICS libraries in the STEPLIB DD statement; see A sample CICS startup job for an example.

Note:

Generally, you do not have to include any Db2 libraries in the DFHRPL DD statement. If you do require Db2 libraries in the DFHRPL concatenation for an application, they must be placed after the CICS libraries.

About this task

The DB2CONN, DB2ENTRY, and DB2TRAN objects describe the global attributes of the CICS Db2 connection, the relationship between CICS transactions and Db2 resources (including application plans and command processors), the attributes of each type of thread, and the type of thread that each transaction can use.

Overview: How you can define the CICS Db2 connection

Defining the CICS Db2 connection involves three different CICS resource definitions: DB2CONN (the Db2 connection definition), DB2ENTRY (the Db2 entry definition), and DB2TRAN (the Db2 transaction definition).

The CICS Db2 connection consists of an overall connection between CICS and Db2, and individual connections known as threads. You can use these resource definitions to define the attributes of the overall connection, and the attributes of the different types of thread. If you have specially defined entry threads for key transactions, you can tell the CICS Db2 attachment facility which CICS transactions can use those threads.

There are several ways to define DB2CONN, DB2ENTRY, and DB2TRAN objects to CICS. You can:

- Define and install DB2CONN, DB2ENTRY, and DB2TRAN objects using the IBM CICS Explorer®.
- Use CICSPlex® SM to define the objects.
- Define and install DB2CONN, DB2ENTRY, and DB2TRAN objects using RDO.
- Define the objects using the CICS CSD update batch utility program DFHCSDUP.

The scope of each object is as follows:

DB2CONN

DB2CONN is the main definition for the CICS Db2 connection. You must install a DB2CONN resource definition before you can start the CICS Db2 connection. You use the DB2CONN definition to define the following items:

- The attributes of the overall CICS Db2 connection:
  - The Db2 subsystem to which CICS connects, or the data-sharing group of Db2 subsystems from which Db2 picks an active member for CICS to connect to.
  - If the connection to Db2 fails, whether CICS reconnects automatically or not, and, if you are using the group attach facility, whether CICS reconnects to the same Db2 subsystem or not.
  - A general-purpose authorization ID for the CICS Db2 attachment facility to sign on to a thread, if no other authorization is needed for the thread.
  - The limit on the total number of TCBs that CICS can use to run threads into Db2 at any one time.
  - How long protected threads are kept before termination.
  - The limit on the number of times a thread can be reused before being terminated.
  - How error messages are communicated to CICS, and what transactions should do if their thread fails.
  - Where messages and statistics are sent.
- The attributes of command threads, which are the special threads used by the DSNC transaction for issuing Db2 commands:
  - The limit on the number of command threads that CICS can use at any one time.
  - What type of authorization ID Db2 checks when a command thread is requested (for example, the ID of the user, the ID of the transaction, and the general-purpose CICS Db2 attachment facility ID).
- The attributes of pool threads, which are the general-purpose threads that are used when transactions do not need a special command or entry thread, or when there are no special threads left for a transaction to use:
  - What type of authorization ID Db2 checks when a pool thread is requested (for example, the ID of the user, the ID of the transaction, the general-purpose CICS Db2 attachment facility ID).
  - What the priority of the thread TCBs is relative to the CICS main TCB.
  - The limit on the number of pool threads that CICS can use at any one time.
  - If a transaction cannot get a pool thread, whether it should wait for one, or be abended.
  - What application plan or dynamic plan exit is used for the pool threads (see Plans, packages and dynamic plan exits).
  - At what point during a transaction's use of the thread Db2 accounting records are produced.
  - If there is a deadlock, whether changes made by the transaction using the thread are rolled back.

You can have only one DB2CONN definition installed in a CICS system at one time, and all the DB2ENTRY and DB2TRAN definitions that you install in the system are associated with the DB2CONN definition. If you discard a DB2CONN definition, all the DB2ENTRY and DB2TRAN definitions are discarded as well. You cannot discard a DB2CONN definition and install another one while CICS is connected to Db2.

Do not confuse the DB2CONN resource definition with the DB2CONN system initialization parameter, which specifies whether you want CICS to start the Db2 connection automatically during initialization.

You can start the CICS Db2 connection with only a DB2CONN definition installed; you do not need any DB2ENTRY and DB2TRAN definitions to make the connection. If you do this, there are no special threads for key transactions (entry threads). All transactions use general-purpose threads from the pool, and the most important transactions have to wait just as long as the least important transactions to get their individual connection into Db2. To ensure that your important transactions are prioritized, create DB2ENTRY and, if necessary, DB2TRAN definitions for them.

DB2ENTRY

You can set up many DB2ENTRY definitions to define different types of entry threads. The entry threads can be used by the transactions that you specify, to gain priority access (or specialized access) to Db2 resources. In effect, you are reserving a certain number of threads that can only be used by those transactions. You can also protect a number of each type of entry thread, which improves performance for heavily-used transactions.

In a DB2ENTRY definition, you can specify a particular transaction, or (by using a wildcard) a group of transactions, that are associated with the DB2ENTRY and can use the type of entry thread that it defines. When those transactions request a thread, the CICS Db2 attachment facility gives them that type of entry thread, if one is available. If you want other transactions to use the same type of entry thread, you can create a DB2TRAN definition for those transactions, which tells CICS that the transactions are associated with a particular DB2ENTRY.

You use each DB2ENTRY definition to define the following items:

- A transaction, or (by using a wildcard in the name) a group of transactions, that can use this type of entry thread.
- The attributes of this type of entry thread:
  - What type of authorization ID Db2 checks when this type of entry thread is requested (for example, the ID of the user, the ID of the transaction, and the general-purpose CICS Db2 attachment facility ID).
  - What the priority of the thread TCBs is relative to the CICS main TCB.
  - The limit on the number of this type of entry thread that CICS can use at any one time.
  - The number of this type of entry thread that are protected for a period of time while they are not being used.
  - If a transaction associated with the DB2ENTRY cannot get this type of entry thread, whether it should wait for this type of entry thread, overflow to use a pool thread, or be abended.
  - What application plan or dynamic plan exit is used for this type of entry thread (see Plans, packages and dynamic plan exits).
  - At what point during a transaction's use of the thread Db2 accounting records are produced.
  - If there is a deadlock, whether changes made by the transaction using the thread are rolled back.

You cannot discard a DB2ENTRY while transactions are using that type of entry thread. You must disable it first, to quiesce activity on the DB2ENTRY, and then discard it. If you discard a DB2ENTRY, the DB2TRANs associated with it are “orphaned”, and the transactions listed in them will use pool threads.

If you are using CECI, the command-level interpreter transaction, to test the syntax of CICS commands in an application program, be aware that in this situation, CICS uses the transaction ID CECI to check for matching DB2ENTRY definitions. If you have set up a DB2ENTRY for your application program and you want to replicate it when using CECI, set up a DB2ENTRY with the same properties for the CECI transaction.

DB2TRAN

You can use DB2TRAN definitions to associate additional transactions with a particular DB2ENTRY. The transactions then use that type of entry thread. Only one DB2TRAN definition can be installed for a specific transaction. In each DB2TRAN definition, you specify the following items:

- The name of a DB2ENTRY.
- A transaction, or (by using a wildcard in the name) a group of transactions, that are associated with the particular DB2ENTRY and can use this type of entry thread.

When those transactions request a thread, the CICS Db2 attachment facility sees that they are associated with the particular DB2ENTRY, and treats them like the transaction or transactions named on the DB2ENTRY definition itself.

Using the Db2 group attach facility

When you are defining the connection between CICS and Db2, you can choose to have CICS connect to a specific Db2 subsystem.

About this task

You can specify the name of this Db2 subsystem using the DB2ID attribute of the DB2CONN definition. However, if you have multiple Db2 subsystems that are using DB2 Version 7 or later, you might want to use the Db2 group attach facility to make it possible for CICS to connect to any of your subsystems, rather than just one named subsystem.

Group attach is a Db2 facility that allows CICS to connect to any one member of a data-sharing group of Db2 subsystems, rather than to a specific Db2 subsystem. The group attach facility chooses any one member of the group that is active on the local MVS image for the connection to CICS (members that are active on other MVS images are not eligible for selection).

To use the group attach facility:

Procedure

1. Activate the group attach facility using the DB2GROUPID attribute of the DB2CONN definition. Specify the group attach name for the group of Db2 subsystems, instead of using the DB2ID attribute to specify the ID of an individual Db2 subsystem.

   With DB2 Version 10, you can also use a subgroup attach name to identify a subset of the group. Group attach means that you can use a common DB2CONN definition, specifying a DB2GROUPID, across multiple cloned AORs, and CICS connects to any active member of that data-sharing group or subgroup. See DB2CONN resources for information about how to define and install a DB2CONN definition.
2. Specify the RESYNCMEMBER attribute of the DB2CONN definition to resolve any indoubt units of work if the connection between CICS and Db2 is broken.

   If the connection is broken, CICS might not reconnect to the same Db2 subsystem - it might choose a different member of the data-sharing group of Db2 subsystems. This means that if indoubt UOWs are being held by the first Db2 subsystem to which CICS connected, they cannot be resolved. See Resolving indoubt units of work (UOWs) for information about the RESYNCMEMBER attribute and how to set it.
3. After the connection has been established, use the **INQUIRE DB2CONN DB2ID()** command to find out which member of the data-sharing group has been chosen for the current connection.
4. If you want CICS to connect to a specific Db2 subsystem, you can override the group attach.

   For example, if you want CICS to connect to the Db2 subsystem with an ID of “xyz”, you can specify the DB2ID using:

   - A **SET DB2CONN DB2ID(xyz)** command
   - A **DSNC STRT xyz** command (see DSNC STRT)

   Each of the previous methods overrides group attach by setting a DB2ID in the installed DB2CONN definition.

Results

Specifying a DB2ID in the previous steps causes the DB2GROUPID attribute of the installed DB2CONN definition to be blanked out. If you want to revert to using group attach, set the DB2GROUPID attribute again using a **SET DB2CONN DB2GROUPID()** command. However, specifying a DB2ID on the **INITPARM**=(DFHD2INI=*db2id*) system initialization parameter does not override group attach. If a DB2GROUPID is set in the DB2CONN definition, the **INITPARM** setting is ignored. See INITPARM system initialization parameter for more information about this parameter.

The MAXOPENTCBS system initialization parameter and TCBLIMIT

As well as the DB2CONN, DB2ENTRY, and DB2TRAN objects you define, the CICS Db2 connection is affected by a system initialization parameter, MAXOPENTCBS. MAXOPENTCBS controls the total number of L8 and L9 mode open TCBs that the CICS region can have in operation at any time.

MAXOPENTCBS is relevant when CICS is connected to Db2 because CICS uses L8 and L9 mode open TCBs to run threads into Db2. L8 and L9 mode open TCBs are reserved for use by task-related user exits that are enabled with the OPENAPI option. These include the CICS Db2 task-related user exit.

In the open transaction environment, the TCBLIMIT attribute of the DB2CONN definition controls how many of the L8 and L9 mode open TCBs can be used by the CICS Db2 task-related user exit to run threads into Db2. If the TCBLIMIT is reached, the CICS Db2 task-related user exit can obtain a TCB from the pool controlled by MAXOPENTCBS, but it must wait before it can use the TCB to run a thread into Db2. When another task stops using its L8 or L9 mode TCB to run a thread into Db2, and so the number of TCBs in use running threads falls below TCBLIMIT, the waiting task is allowed to use its own L8 or L9 mode TCB to run a thread into Db2. However, if MAXOPENTCBS is reached, no more L8 and L9 mode open TCBs are allowed in the CICS region, and the CICS Db2 task-related user exit cannot even obtain an L8 or L9 mode TCB for its use. It must wait until an L8 mode TCB or an L9 mode TCB (used by OPENAPI programs) is released by another task and returned to the pool controlled by MAXOPENTCBS. If an L8 TCB is released, the task can use this TCB, if it was an L9 TCB that was released then CICS "steals" the TCB, meaning it detaches the L9 TCB and attaches an L8 TCB in its place for the task to use.

To ensure that you have enough L8 and L9 mode open TCBs available to meet your Db2 workload, set the limit in your MAXOPENTCBS system initialization parameter to a value greater than the limit set in the TCBLIMIT attribute of your DB2CONN definition. If MAXOPENTCBS is lower than TCBLIMIT, the system can run out of L8 and L9 mode open TCBs before it reaches TCBLIMIT. When CICS connects to Db2, a warning message, DFHDB2211, is issued if the CICS Db2 attachment facility detects that the setting of MAXOPENTCBS in the SIT is lower than the TCBLIMIT setting in the DB2CONN definition. If you receive this warning message, adjust your MAXOPENTCBS limit.

In addition, when running with Transaction Isolation active, set MAXOPENTCBS to the value of max tasks (MXT) or higher. This setting minimizes the possibility of TCB stealing due to a TCB being allocated to the wrong subspace.

What happens during SQL processing

The main Db2 activities involved in processing CICS transactions with SQL calls are: thread creation, SQL processing, commit processing, thread release, and thread termination.

These main activities are performed for each transaction. The exact work involved in each activity depends on the DB2CONN and DB2ENTRY options, the BIND options, and whether packages are used.

Thread creation

At thread creation time a number of activities occur, these activities include the plan authorization check, which must always happens.

The following activities can occur at thread creation time, depending on the BIND options that have been specified:

- 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.

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.

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

For each SQL statement that is processed there are a number of activities that can occur. These activities vary depending on thread reuse and the BIND options.

- For the first SQL call in a transaction reusing a thread with a new authorization ID the following actions can occur:
  - Signon
  - 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.
- Acquire referenced TS locks, if not already taken.
- Load DBDs in the EDM pool, if they are not already there.
- Process the SQL statement.

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 binding plans.

Commit processing

At commit time a number of activities can occur, these activities include the releasing of page locks and TS locks.

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.

Thread release

Transactions release the thread they are using at different times depending on whether the transaction is terminal-oriented and the options that have been specified.

If the transaction is terminal-oriented, or if it is non-terminal-oriented [1] 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(USE) 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.
- Certain Db2 modifiable special registers are not at their initial value. SQL: The language of Db2 in Db2 for z/OS product documentation lists these special registers.
- 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(USE) and RELEASE(DEALLOCATE) give the same advantages as for the terminal-oriented transactions with many syncpoints.

Note: 1 Non-terminal-oriented transactions are transactions that are not associated with a terminal.

Thread termination

At thread termination time a number of activities can occur, these activities include the release of TS locks and freeing work storage.

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.

How threads are created, used, and terminated

Three main types of threads are used by the CICS Db2 attachment facility: command threads, entry threads, and pool threads. There are specific rules that apply to thread creation, use, and termination.

When CICS is connected to Db2, the thread TCB is the open TCB on which the CICS Db2 attachment facility is already running. When it needs to run a thread, the CICS Db2 attachment facility associates the open TCB with the Db2 connection control block and thread it wants to use, and the open TCB then runs the thread. When the thread is no longer needed, the open TCB dissociates from it, and the Db2 connection control block and thread become available for reuse by another open TCB.

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

- When CICS is connected to Db2 and is using open TCBs as the thread TCBs, before an SQL request can be passed to Db2 a thread must be available for the transaction. The open TCB associates itself with the thread, and becomes the thread TCB until it dissociates from the thread.
- 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 typically releases the thread at sync point and end-of-task. The thread is *not* released at sync point 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 end-of-task only, unless NONTERMREL=YES is specified in the DB2CONN.
- When a transaction releases a thread, the thread can be reused by another transaction that is specifying the same plan and that is defined in the same DB2ENTRY. Pool threads can be reused by any waiting (queued) transaction that is specifying the same plan and using a pool thread.
- An unprotected thread is terminated immediately it is released, unless another transaction is waiting (queued) for the thread.

- A protected thread is terminated if it is not used during two consecutive purge cycles. With default settings, this averages about 45 seconds. This value can be modified by the PURGECYCLE parameter of the DB2CONN.
- A thread is canceled and terminated if it is active in Db2 at the time the CICS task that is using the thread is purged or forcepurged.
- The REUSELIMIT parameter of the DB2CONN specifies the maximum number of times a thread can be reused before it is terminated. The reuse limit applies to unprotected threads both in the pool and on a DB2ENTRY, and to protected DB2ENTRY threads.
- The THREADWAIT parameter defines whether the requests for a thread should be queued, abended, or sent to the pool thread in the case of a shortage of entry or command threads. If THREADWAIT=YES is specified instead of THREADWAIT=POOL the transaction is queued rather than sent to the pool thread. Using THREADWAIT=YES avoids the thread initialization and termination overhead. If a transaction is made to wait because of the lack of entry threads, the CICS Db2 attachment facility queues the transaction. The advantages of this queuing are that, when the entry thread finishes its current piece of work, it continues with the next transaction immediately.
- TCBLIMIT specifies the maximum number of TCBs that can be used to run Db2 threads, which, in turn, limits the maximum number of active Db2 threads. THREADLIMIT specifies the maximum number of active Db2 threads. THREADLIMIT is modified dynamically. CTHREAD is specified in ZPARMS, and it defines the number of concurrent threads for all of Db2. The sum of all the active threads from TSO users, all CICS and IMS systems and other systems accessing Db2 should not exceed CTHREAD. Otherwise, the result could be unpredictable response times. When this occurs, a CICS Db2 attachment facility "create thread" request is queued by Db2, and the CICS transaction is placed is a wait state until a thread is available.

  CICS manages the total number of L8 and L9 mode TCBs that the CICS region has in operation at any one time. The maximum number of L8 and L9 mode TCBs in the open TCB pool is controlled by either the MXT or MAXOPENTCBS system initialization parameters. The MAXOPENTCBS system initialization parameter, if specified, sets the value for the open TCB pool. If the MAXOPENTCBS system initialization is not specified, CICS sets the limit for the L8 and L9 mode open TCB pool automatically based on the maximum number of tasks specified for the CICS region (the MXT value), using the formula `(2 * MXT Value) + 32`. In the open transaction environment, TCBLIMIT controls how many of these open TCBs can be used by the CICS Db2 task-related user exit to run threads into Db2. If the limit set by CICS is reached, no more open TCBs are allowed in the CICS region, and the CICS Db2 task-related user exit cannot obtain an open TCB for its use.

  Each thread linking CICS to Db2 has a corresponding TCB in the CICS address space. Too many TCBs per address space involve the MVS dispatcher scanning the TCBs to identify an active TCB. If there are a large number of TCBs, there could be a significant cost of processor time. However, if you have too few TCBs available to meet your Db2 workload, transactions must wait to obtain a TCB.

  Increasing the TCBLIMIT value or setting up an additional CICS system with access to the same Db2 system may require increasing the CTHREAD parameter of Db2.

  For a protected entry thread environment, review the number of application plans and, if possible, reduce the number of plans by combining infrequently used ones while balancing the issues of plan size and security. Initially, you should start with one thread per plan. In a high-volume transaction processing environment, you can estimate the initial number by using the occupancy time of a thread by a transaction and multiplying it with the expected transaction rate. For example, an occupancy time of 0.2 seconds and a transaction rate of 20 transactions per second (0.2 x 20) would give an initial thread number of between three and four.

Protected entry threads

Protected entry threads are defined with the following DB2ENTRY parameters: `PROTECTNUM(n)` and `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
When CICS is connected to Db2, and is using the open transaction environment, a new or existing open TCB is allocated to the CICS task before calling the CICS Db2 attachment facility. If the limit set automatically by CICS for the number of L8 and L9 mode open TCBs is reached, no more open TCBs can be created, and the task enters a CICS dispatcher wait until an open TCB is available. At the end of the task, the open TCB is returned to the pool of open TCBs managed by the CICS dispatcher.

Thread creation
A thread is created only if an existing thread is not available. If no thread is available for a task, a new thread is created and related to the task's open TCB, provided THREADLIMIT is not exceeded. If TCBLIMIT is reached, no more open TCBs can be used as thread TCBs for the DB2ENTRY.

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 purge cycles.

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. When a transaction overflows to the pool, the transaction is now controlled by the PRIORITY, THREADLIMIT, and THREADWAIT attributes that are specified for pool threads in the DB2CONN definition, and those attributes in the DB2ENTRY definition are ignored. The remaining attributes that are specified in the DB2ENTRY definition for the entry thread still apply to the transaction, that is, ACCOUNTREC, AUTHID or AUTHTYPE, DROLLBACK, and PLAN or PLANEXITNAME. The PROTECTNUM attribute in the DB2ENTRY definition is no longer relevant for a transaction that overflows to the pool, so this setting is ignored.

Unprotected entry threads for critical transactions

Unprotected entry threads for critical transactions are defined with the following DB2ENTRY parameters: `PROTECTNUM(0)` and `THREADLIMIT(n)`.

This is the recommended type of definition for:

- Critical transactions requiring a fast response time, 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.

Unprotected entry threads for critical transactions are created, used, and terminated as follows:

TCB attach
No thread TCBs are attached when the CICS Db2 attachment facility is started.

A TCB is attached only if needed by a thread.

Thread creation
A thread is created only when needed by a transaction.

When CICS is connected to Db2 (and so is using the open transaction environment), if no thread is available, but an open TCB can be used as a thread TCB for this DB2ENTRY, a new thread is created

and related to the TCB, provided THREADLIMIT is not exceeded. If TCBLIMIT is reached, no more open TCBs can be used as thread TCBs for the DB2ENTRY.

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. When a transaction overflows to the pool, the transaction is now controlled by the PRIORITY, THREADLIMIT, and THREADWAIT attributes that are specified for pool threads in the DB2CONN definition, and those attributes in the DB2ENTRY definition are ignored. The remaining attributes that are specified in the DB2ENTRY definition for the entry thread still apply to the transaction, that is, ACCOUNTREC, AUTHID or AUTHTYPE, DROLLBACK, and PLAN or PLANEXITNAME. The PROTECTNUM attribute in the DB2ENTRY definition is no longer relevant for a transaction that overflows to the pool, so this setting is ignored.

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

Unprotected entry threads for background transactions

Unprotected entry threads are defined with the following DB2ENTRY parameters: `PROTECTNUM(0)`, `THREADLIMIT(0)`, and `THREADWAIT(POOL)`.

Unprotected entry threads are recommended for low volume transactions that do not require a fast response time. All transactions are forced to use pool threads.

Unprotected entry threads for background transactions are created, used, and terminated as follows:

TCB attach
No subtask thread TCB is ever attached for this thread definition because THREADLIMIT=0. A pool thread TCB (or, in the open transaction environment, an open 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

Pool threads are defined with the `THREADLIMIT(n)` DB2CONN parameter.

There are four situations where 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 for the pool.

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 Db2 attachment facility is started.

A TCB is attached only if needed by a thread.

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.

Selecting thread types for optimum performance

To optimize performance, select the thread type to use for a set of transactions together with the BIND parameters for the corresponding plan.

Selecting the thread type and BIND parameters together is important because the BIND parameters determine whether a number of activities are related to the thread or to the transactions. For more information, see “Selecting BIND options for optimum performance” on page 22.

The following types of thread are available:

Protected entry threads
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)

For more information, see “How threads are created, used, and terminated” on page 17.

Unprotected entry threads for critical transactions
Unprotected entry threads for critical transactions are recommended for:

- Critical transactions requiring a fast response time, 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.

Unprotected entry threads for background transactions
Unprotected entry threads for background transactions are recommended for transactions with low volume that do not require a fast response time. All transactions are forced to use pool threads.

Using protected threads (by specifying PROTECTNUM=n on the DB2ENTRY definition for an entry thread) is a performance option that reduces the resources involved in creating and terminating a thread. A protected thread is not terminated when a transaction ends, and the next transaction associated with the same DB2ENTRY reuses the thread. By using protected threads, you eliminate much of the work required for thread creation and termination for individual transactions. For performance reasons, it is recommended that you use protected entry threads whenever possible, and especially where a

transaction is frequently used, or issues many SYNCPOINTS. The main exception is when a transaction is infrequently used, because even a protected thread is likely to terminate between such transactions.

From an accounting viewpoint, the situation is different. An accounting record is produced for each thread termination and for each new user signon. 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 Accounting and monitoring in a CICS Db2 environment in Monitoring for more information about accounting in a CICS Db2 environment.

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.

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 to use the same DB2ENTRY 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.

It is important that you coordinate your DB2CONN, DB2ENTRY, and BIND options. For more information, see “Coordinating your DB2CONN, DB2ENTRY, and BIND options” on page 22.

Selecting BIND options for optimum performance

You must coordinate the BIND options that you select for a plan with the thread types used by the transactions associated with that plan.

For an overview of the bind process, see The bind process. For advice on selecting thread types, see “Selecting thread types for optimum performance” on page 21.

The bind option VALIDATE can affect performance. Use VALIDATE(BIND) in a CICS environment. For more information, see Bind options and considerations for programs.

It is important that you coordinate your DB2CONN, DB2ENTRY, and BIND options. For more information, see “Coordinating your DB2CONN, DB2ENTRY, and BIND options” on page 22.

Coordinating your DB2CONN, DB2ENTRY, and BIND options

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

One of the most important things that you are required to do to optimize performance is to define whether a given set of transactions must use one or more protected threads. “Selecting thread types for optimum performance” on page 21 has more detailed advice about this. Next; consider defining the BIND parameters to minimize the total amount of work related to the main activities involved in processing SQL transactions. See “Selecting BIND options for optimum performance” on page 22 for more information.

In general, you are recommended to set the initial values for your DB2CONN, DB2ENTRY, and BIND options to the values shown in Table 1 on page 22. You might 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 for whether transactions overflow to the pool.

Table 1. Recommended combinations of DB2CONN, DB2ENTRY and BIND options

Transaction Description	**Thread Type**	**Overflow**	**ACQUIRE**	**RELEASE**
High volume (all types)	Protected Entry	Note 1	USE	DEALLOCATE

Table 1. Recommended combinations of DB2CONN, DB2ENTRY and BIND options (continued)

Transaction Description	Thread Type	Overflow	ACQUIRE	RELEASE
Terminal-oriented with many commits (plus non-terminal if NONTERMREL=YES)	Protected Entry	Note 2	USE	DEALLOCATE
Low volume, requires fast response time	Unprotected Entry	Yes	USE	COMMIT
Low volume, limited concurrency	Unprotected Entry	Never	USE	COMMIT
Low volume, does not require fast response time	Pool	Not Applicable	USE	COMMIT
Non-terminal-oriented with many commits (NONTERMREL=NO)	Note 3	Note 3	USE	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. Threads are held until EOT. Use pool threads for a short transaction. Consider entry threads for longer running transactions.

In Table 1 on page 22 limited concurrency means only a limited number (n) of transactions are allowed to run at the same time. A special case exists when n=1, in this 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 are recommended to 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 22 shows, a few combinations of DB2CONN, DB2ENTRY, and BIND options are generally recommended. However, in specific situations other combinations can be used.

Table 2 on page 24 shows a summary of the activities involved in processing SQL requests for the three recommended sets of DB2CONN, DB2ENTRY, and BIND specifications. The table also demonstrates the performance advantage of using protected threads without changing the authorization ID. Required activities are marked as follows in the table:

X
Required activity

(1)
Required only if new authorization ID

(2)
Required only if SQL section is not already in EDM pool

(3)
Required only if SQL section is not already in Cursor Table

Table 2. Activities involved in processing SQL requests for different DB2CONN, DB2ENTRY, and BIND specifications

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

Chapter 3. Administering with CICS Db2

This information discusses operating the CICS Db2 attachment facility.

Starting the CICS Db2 attachment facility

The CICS Db2 attachment facility can be started automatically at initialization, or manually.

About this task

Automatic connection at CICS initialization

You can configure CICS, by any one of the following methods, so that at initialization, an automatic connection between CICS and Db2 is established:

- 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

You can manually establish the connection between CICS and Db2 by any one of the following methods:

- Using a **DSNC STRT** command
- Using a **CEMT SET DB2CONN CONNECTED** command
- Running a user application that issues an **EXEC CICS SET DB2CONN CONNECTED** command

Stopping the CICS Db2 attachment facility

The CICS Db2 attachment facility, and hence the CICS-Db2 connection, can be quiesce stopped or force stopped.

About this task

A quiesce stop waits for all CICS transactions currently using Db2 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 and shuts down the CICS Db2 connection.

About this task

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 canceled, no shutdown of the attachment facility is initiated by the TRUE.

Manual disconnection

You can disconnect CICS from Db2 in a number of ways.

About this task

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 40.
- 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.

Resolving indoubt units of work (UOWs)

Indoubt units of work (UOWs) can occur when CICS, Db2, or the whole system fails while a transaction is carrying out syncpoint processing, that is, during processing of an **EXEC CICS SYNCPOINT** or **EXEC CICS RETURN** command. Indoubt UOWs are typically resolved when the connection between CICS and Db2 is reestablished.

If more than one recoverable resource is involved, CICS 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.

In situations where only Db2 and a single CICS system are involved with the unit of work, CICS is always the coordinator. If further parties are involved, for example, by means of a LU6.2 communication link, it is possible that a remote CICS system might be the overall coordinator of the unit of work instead of the local CICS system. In this situation, it is possible that both Db2 and the local CICS system are indoubt about the outcome of the UOW. If a failure occurs in this situation, the local CICS system might shunt the unit of work, depending on the definition for the transaction. The unit of work is then considered to be shunted indoubt.

Indoubt UOWs are typically 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 was backed out or committed. If a UOW is shunted indoubt, this exchange of information is deferred until the remote coordinator has informed CICS of the outcome. However, if you are using group attach, CICS might reconnect to a different Db2 subsystem, and be unable to exchange information about the indoubt UOWs held by the previously connected Db2 subsystem. The RESYNCMEMBER attribute of the DB2CONN definition is used to solve this problem.

Resolving indoubt UOWs when using group attach

If you are using group attach and the connection between CICS and Db2 is broken, CICS might not reconnect to the same Db2 subsystem—it might choose a different member of the data-sharing group of

Db2 subsystems. This means that if indoubt UOWs are being held by the first Db2 subsystem to which CICS connected, they cannot be resolved.

To solve this problem, CICS maintains a history of the last Db2 data-sharing group member to which it connected, which is cataloged and maintained across warm, emergency and cold starts (but not initial starts). During connection or reconnection to Db2, the CICS Db2 attachment facility checks this history to see if any outstanding UOW information is being held for the last Db2 data-sharing group member to which it connected, and acts as follows:

- If no outstanding UOW information is being held and if CICS has not performed an emergency restart, group attach operates normally and chooses any active member of the data-sharing group for the connection. If CICS has performed an emergency restart, CICS attempts to connect to the member that it was last connected to.
- If outstanding UOW information is being held, the next action depends on the setting you have chosen for the RESYNCMEMBER attribute of the DB2CONN definition.
  - If RESYNCMEMBER is set to YES, indicating that you require resynchronization with the last recorded Db2 data-sharing group member, CICS ignores the group attach facility, and the CICS Db2 attachment facility waits until it can reconnect to that last connected Db2 data sharing group member, to resolve the indoubt units of work. UOWs which are shunted indoubt are not included in this process, because CICS itself is unable to resolve those UOWs at this time. Resynchronization for those UOWs will occur when CICS has resynchronized with its remote coordinator.
  - If RESYNCMEMBER is set to NO, perhaps because you want to reconnect as fast as possible, CICS makes one attempt to reconnect to the last recorded Db2 data-sharing group member. If this attempt is successful, the indoubt UOWs (with the exception of UOWs that are shunted indoubt) can be resolved. If it is unsuccessful, CICS uses group attach to connect to any active member of the Db2 data-sharing group, and the warning message DFHDB2064 is issued stating that there may be unresolved indoubt UOWs with the last recorded member.

See DB2CONN resources for information on setting RESYNCMEMBER. The RESYNCMEMBER option can also be set using a **SET DB2CONN RESYNCMEMBER** command.

Resolving indoubt units of work using Db2 restart-light

CICS supports the enhanced Db2 restart-light capability provided in DB2 Version 8.

Restart-light mode is intended for a cross-system restart in the event of an MVS system failure, where the CICS systems are configured to use group attach (see Using the Db2 group attach facility). In DB2 Version 7, the Db2 restart-light capability allowed a Db2 data-sharing member to restart with a minimal storage footprint, free retained locks, and then terminate normally. However, this only applied to in-flight units of work, and not indoubt units of work. For DB2 Version 8, this capability was extended to encompass indoubt units of work. This enables a CICS system to connect to a Db2 restart-light subsystem for the purpose of resynchronizing indoubt units of work. This capability is especially useful because Db2 does not support peer recovery; that is, one Db2 subsystem cannot resynchronize indoubt units of work on behalf of another Db2 subsystem.

In a typical scenario, if an MVS LPAR fails, a cross-system restart can be initiated which involves restarting the failed CICS systems on another LPAR, and bringing up the failed Db2 subsystem on that LPAR in restart-light mode. Assume that the CICS systems have been configured to use group attach, with RESYNCMEMBER(YES) and STANDBYMODE(RECONNECT) on the DB2CONN definition, and there are indoubt units of work outstanding in Db2. The Db2 restart-light subsystem initializes, frees retained locks for any in-flight UOWs, and then awaits resynchronization with CICS for the indoubt UOWs. Each CICS system initializes, and detects whether it has outstanding units of work and whether RESYNCMEMBER(YES) has been specified. Where both these conditions are true, the CICS system ignores group attach and reconnects back to the last Db2 subsystem, which is the Db2 restart-light subsystem. The indoubt UOWs are now resynchronized, but no new transactions are allowed to access Db2. When all the indoubt UOWs have been resolved in the Db2 restart-light subsystem, it terminates. Because the CICS systems are defined with STANDBYMODE(RECONNECT), when the Db2 restart-light subsystem terminates, they drop into standby mode and attempt a reconnection to Db2. Now, because all the

indoubt units of work have been resolved, RESYNCMEMBER does not apply and group attach can be used. The CICS systems will reconnect to a normal, active Db2 subsystem.

Recovery of resynchronization information for indoubt UOWs

CICS maintains information about UOWs needed for resynchronization on its system log. Resynchronization information is maintained by CICS across warm, emergency, and cold starts.

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 want to reinstall resources from the CICS system definition data set (CSD), use a cold start, 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 resynchronization is 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.

Managing the CICS Db2 attachment facility

You can manage 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.

About this task

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. See CEMT INQUIRE DB2CONN and CEMT SET DB2CONN.

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

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

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 command 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 33.

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 34.

The DSNC DISP command also displays CICS Db2 statistics to a CICS terminal. These statistics are only a subset of the CICS Db2 statistics that you can obtain using the CICS COLLECT STATISTICS and PERFORM STATISTICS commands. These statistics are subject to the same resetting characteristics as all CICS statistics.

For more information on the DSNC DISP STAT command, see “DISPLAY STATISTICS output” on page 37. For more information on the full CICS Db2 statistics available, see CICS Db2 statistics in Reference.

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 38.

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.

About this task

Commands that use the DSNC transaction to route commands to the Db2 subsystem 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 4 on page 29 shows the CICS Db2 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 the DSNC transaction” on page 32.

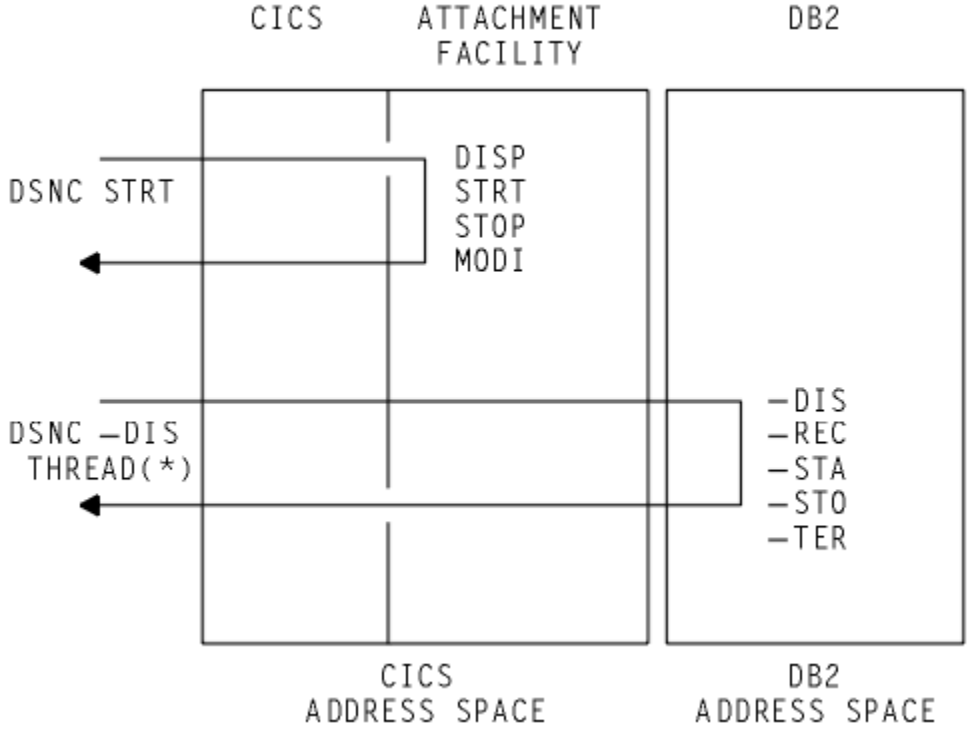


Figure 4. Examples of CICS Db2 attachment facility commands and some Db2 commands

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 signon. 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.

About this task

The SMF records produced by Db2 can be directed to an SMF data set in the following ways:

- Specifying at Db2 installation that accounting or statistics data or both are required.

  To do this the MON,SMFACCT or SMFSTAT of the initialization macro DSN6SYSP are set to YES. See Db2 for z/OS product documentation for 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 performance 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, you can use Db2 to send this data to a generalized trace facility (GTF). Db2 provides -START TRACE and -STOP TRACE commands which you can issue from a CICS terminal using the DSNC transaction.

About this task

Use the following commands with the DSNC transaction to specify the trace scope, trace destination, and trace constraints:

Trace scope
: Set this to `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.

CICS-supplied transactions for CICS Db2

CICS provides a number of transactions that you can use to manage the interface with Db2.

CEMT

The following CEMT options are available:

- CEMT INQUIRE DB2CONN
- CEMT SET DB2CONN

- CEMT INQUIRE DB2ENTRY
- CEMT SET DB2ENTRY
- CEMT INQUIRE DB2TRAN
- CEMT SET DB2TRAN

The equivalent **EXEC CICS INQUIRE** and **EXEC CICS SET** commands are described in System commands.

PLAN and PLANEXITNAME options are available on the **INQUIRE DB2TRAN** command, so you can find out in a single step which plan is used by a specified transaction or set of transactions, or which transactions use a specified plan.

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).

The DSNC transaction executes program DFHD2CM1, which handles both CICS Db2 attachment facility and Db2 commands. You can distinguish Db2 commands from CICS Db2 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 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.

DFHD2CM1 can also be activated by transactions other than DSNC. Thus you can give a different transaction code to each CICS Db2 attachment facility command, and to each Db2 command. This enables you to apply different levels of security to each command.

Alternative transaction definitions for CICS Db2, supplied in sample group DFH$DB2

Alternative transaction definitions for CICS Db2 attachment facility commands are supplied with the following names:

- DISC
- DISP
- STRT
- STOP
- MODI

Alternative transaction definitions for Db2 commands are also supplied with the following names:

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

Issuing commands to Db2 using the DSNC transaction

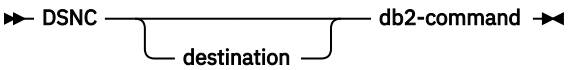
The DSNC transaction allows you to enter Db2 commands from CICS.

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 more information about CICS security, see Security in a CICS Db2 environment.

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`
BMS paging support can be used to support the scrolling of DSNC Db2 commands from your terminal. For example specify SIT keywords:

PGCHAIN=X/,

PGCOPY=C/,

PGPURGE=T/,

PGRET=P/,

SKRPF7='P',

SKRPF8='N',

For further information about the SIT keywords and parameters, see System initialization parameter descriptions and summary.

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 TOKEN
IYK4Z2G1 N        3              JTILLI1           00BA     0
IYK4Z2G1 T        3 ENTRXC060001 CICSUSER TESTC06  00BA    16
IYK4Z2G1 T        3 POOLXP050002 CICSUSER TESTP05  00BA    17
IYK4Z2G1 T  *     6 COMDDSNC0003 JTILLI1           00BA    18
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I : DSNVDT '-DIS THREAD' NORMAL COMPLETION
DFHDB2301 07/09/98 13:36:36 IYK4Z2G1 DSNC DB2 command complete.
```

Figure 5. Sample output from DSNC -DISPLAY command

DSNC DISCONNECT

Use the CICS Db2 attachment facility command **DSNC DISCONNECT** to disconnect threads. This 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 CICS transaction attach security check for transaction DSNC and the CICS command security check for resource DB2CONN. This command requires READ access.

For more information about CICS security, see Security in a CICS Db2 environment.

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.

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 might 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 6. Sample output from DSNC -DISCONNECT command

DSNC DISPLAY

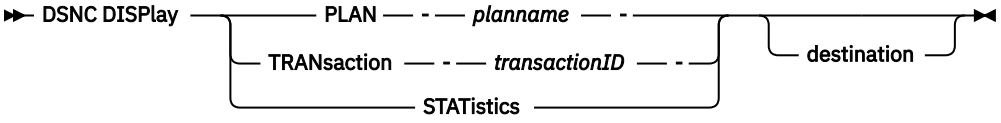
Use the CICS Db2 attachment facility command **DSNC DISPLAY** to display information about active CICS Db2 threads and the CICS transaction using them, or to display statistical information associated with DB2ENTRYs and the DB2CONN resource.

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 CICS transaction attach security check for transaction DSNC and the CICS command security check for resource DB2CONN. This command requires READ access.

For more information about CICS security, see Security in a CICS Db2 environment.

Parameter description

PLAN *planname*

Displays information about the threads for *planname*. *Planname* is a valid plan name for which information is displayed.

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

For example, the following command displays information on all active CICS Db2 plan IDs and sends the display information to another terminal designated as MTO2.

```
DSNC DISP PLAN * MTO2
```

Tran *transactionID*

Contains the transaction ID for which thread information is displayed. *transactionID* is 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.

For example, the following command displays information about all active CICS Db2 transactions.

```
           DSNC DISP TRAN
```

STAT

Displays the statistical counters associated with each CICS Db2 resource definition. The counters concern the usage of the available connections of the CICS Db2 attachment facility to Db2.

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.

destination

Specify a different terminal to receive the requested information. *destination* is 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
```

Usage notes

If you issue this command from CICS while the CICS Db2 attachment facility is active but the Db2 subsystem is not, a statistics display is produced with no obvious indication that the subsystem is not operational.

Message DFHDB2037 appears in the CICS message log to indicate that the attachment facility is waiting for Db2 to start.

DISPLAY PLAN or TRAN output

The output of the **DSNC DISPLAY PLAN** or **DSNC DISPLAY TRANSACTION** command includes the name of the DB2ENTRY, *POOL, or *COMMAND for the DSNC command call.

Figure 7 on page 36 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, *POOL for the pool, or the *COMMAND for the DSNC command call.

```
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     * 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 7. Sample output from DSNC DISPLAY (PLAN or TRANSACTION) command

The column named 'S' denotes the status of the thread, and can take the following values:

*****
: The thread is active within a unit of work, and is currently executing in Db2.

A
: The thread is active within a unit of work, but is not currently executing in Db2.

I
: The thread is inactive; it is a protected thread that is waiting for new 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

This topic describes the output of a **DSNC DISPLAY STATISTICS** command.

```
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 8. 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 signon invocations for transactions associated with this entry. A signon does not indicate whether a new thread is created or an existing thread is reused. If the thread is reused, a signon 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 is set to POOL, W/P indicates the number of times the transaction overflowed to the pool. An overflow to the pool only shows up in the statistics for the individual DB2ENTRY, and is not reflected in the pool statistics.

If THREADWAIT is set to YES, this reflects the number of times that either the transaction had to wait for a thread (because the number of active threads had reached THREADLIMIT), or the transaction could not use a new thread TCB (because the number of TCBs in use running threads had reached TCBLIMIT).

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-water mark of threads.

Note: In releases of CICS before CICS Transaction Server for OS/390®, Version 1 Release 2, the HIGH value also included the transactions forced to wait for a thread or those diverted to the pool. From CICS Transaction Server for OS/390, Version 1 Release 2 onwards, the HIGH value *only* represents threads 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). 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

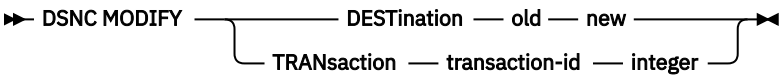
Use the CICS Db2 attachment facility command DSNC MODIFY to modify: the message queue destinations of the DB2CONN, the THREADLIMIT value for the pool, DSNC commands, or DB2ENTRYs. These modifications can also be made using CEMT, or EXEC CICS, SET DB2CONN or DB2ENTRY commands.

Environment

This command can be issued only from a CICS terminal.

Syntax

MODIFY syntax



Abbreviation

DSNC MODI or MODI (using the MODI transaction from the CICS Db2 sample group DFH$DB2).

Authorization

You can control access to this command using 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 Security in a CICS Db2 environment.

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 *transaction-ID integer*
Specifies that the THREADLIMIT value associated with the given transaction or group is to be modified. 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:

```
           DSNC MODI DEST MTO1 MTO2
```

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

Figure 9. 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 10. 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 11. 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 12. Sample output from DSNC MODIFY TRANSACTION command (changing DB2ENTRY thread limit)

DSNC STOP

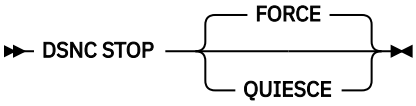
Use the CICS Db2 attachment facility command **DSNC STOP** to stop the attachment facility. Alternatively, you can issue a **CEMT** or **EXEC CICS SET DB2CONN NOTCONNECTED** command to stop the attachment facility.

Environment

This command can be issued only from a CICS terminal.

Syntax

STOP 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 CICS transaction attach security check for transaction DSNC, and the CICS command security check for resource DB2CONN. This command requires UPDATE access.

This command requires UPDATE access. For more information about CICS security, see Security in a CICS Db2 environment.

Parameter description

QUIESCE
Specifies that the CICS Db2 attachment facility is to be stopped after CICS transactions currently running complete. QUIESCE waits for all active transactions to complete, so new UOWs can start and acquire threads.

FORCE
Specifies that the CICS Db2 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 13. Sample output from DSNC STOP command

The message resulting from the DSNC STOP command shown in Figure 13 on page 41 is replaced by the message shown in Figure 14 on page 41 when shutdown is complete.

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

Figure 14. 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 DF2D is proceeding.
```

Figure 15. Sample output from DSNC STOP FORCE command

The message resulting from the DSNC STOP FORCE command shown in Figure 15 on page 41 is replaced by the message shown in Figure 16 on page 41 when shutdown is complete.

```
DFHDB2025I 07/09/98 15:10:55 IYK4Z2G1 The CICS-DB2 attachment has disconnected
from DB2 subsystem DF2D group DFP2
```

Figure 16. Sample output from DSNC STOP FORCE when shutdown is complete

In this example, group attach was used, so the name of the Db2 subsystem and the name of its group are shown.

DSNC STRT

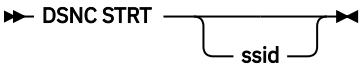
Use the **DSNC STRT** command to start the CICS Db2 attachment facility. The CICS Db2 attachment facility allows CICS application programs to access Db2 databases. Alternatively, issue a **CEMT** or **EXEC CICS SET DB2CONN CONNECTED** command to allows your CICS application programs to access Db2 databases.

Environment

This command can be issued only from a CICS terminal.

Syntax

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 CICS transaction attach security check for transaction DSNC, and the CICS command security check for resource DB2CONN. This command requires UPDATE access.

For more information about CICS security, see Security in a CICS Db2 environment.

Parameter description

ssid
: Specifies a Db2 subsystem ID to override the Db2 subsystem ID (DB2ID) or Db2 data-sharing group ID (DB2GROUPID) specified in the DB2CONN. You cannot specify a Db2 data-sharing group ID in a DSNC STRT command.

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. Resource definitions must be installed from the CICS system definition data set (CSD) before attempting to start the CICS Db2 attachment facility.

If you issue a DSNC STRT command and specify a Db2 subsystem ID, any DB2GROUPID in the installed DB2CONN definition is blanked out, and needs to be set again (using CEDA INSTALL or a SET DB2CONN command) to use group attach on subsequent occasions.

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 DB2GROUPID in the installed DB2CONN for group attach.
4. Use the subsystem ID if specified on the INITPARM when the DB2ID and DB2GROUPID in the last installed DB2CONN are blank (or have subsequently been set to blanks). On any startup, INITPARM is always used if the last installed DB2CONN contained a blank DB2ID and a blank DB2GROUPID, even if the DB2ID or DB2GROUPID were subsequently changed using a SET command.
5. Use a default subsystem ID of DSN.

Examples

Example 1

To start the CICS Db2 attachment facility using the Db2 subsystem ID (DB2ID) or Db2 data-sharing group ID (DB2GROUPID) from an installed DB2CONN:

```
                    DSNC STRT
```

In this example, group attach is used, so the name of the Db2 subsystem and the name of its group are shown.

```
DFHDB2023I 07/09/98 15:06:07 IYK4Z2G1 The CICS DB2 attachment has connected to
DB2 subsystem DF2D group DFP2
```

Figure 17. 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 (DB2ID) or Db2 data-sharing group ID (DB2GROUPID) in the DB2CONN with the Db2 subsystem ID DB3A:

```
                  DSNC STRT DB3A
```

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

Figure 18. Sample output from DSNC STRT using DB3A

If you are not using group attach, and the Db2 subsystem 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 19. Sample output from DSNC STRT when Db2 is not active and STANDBYMODE=NOCONNECT

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

```
DFHDB2037 07/09/98 15:15:42 IYK4Z2G1 DB2 subsystem DB3A is not active.
The CICS DB2 attachment facility is waiting.
```

Figure 20. Sample output from DSNC STRT when Db2 is not active and STANDBYMODE=CONNECT or RECONNECT

If you are using group attach, and no Db2 subsystems in the data-sharing group are 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:

```
DFHDB2037 07/09/01 12:30:10 IYK2ZFV1 DB2 group DFP2 has no active members.
```

Figure 21. Sample output from DSNC STRT with group attach when no Db2 subsystems in the data-sharing group are active and STANDBYMODE=NOCONNECT

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

```
DFHDB2037 07/09/01 12:55:00 IYK2ZFV1 DB2 group DFP2 has no active members.
The CICS DB2 attachment facility is waiting.
```

Figure 22. Sample output from DSNC STRT with group attach when no Db2 subsystems in the data-sharing group are active and STANDBYMODE=CONNECT or RECONNECT

Chapter 4. Security for Db2

In the CICS Db2 environment, there are four main stages at which you can implement security checking.

The four stages are:

- When a CICS user signs on to a CICS region. CICS sign-on authenticates users by checking that they supply a valid user ID and password..
- When a CICS user tries to use or modify a CICS resource that is related to Db2. This could be a DB2CONN, DB2ENTRY or DB2TRAN resource definition; or a CICS transaction that accesses Db2 to obtain data; or a CICS transaction that issues commands to the CICS Db2 attachment facility or to Db2 itself. At this stage, you can use CICS security mechanisms, which are managed by RACF® or an equivalent external security manager, to control the CICS user's access to the resource.
- When the CICS region connects to Db2, and when a transaction acquires a thread into Db2. Both the CICS region and the transaction must provide authorization IDs to Db2, and these authorization IDs are validated by RACF or an equivalent external security manager.
- When a CICS user tries to use a CICS transaction to execute or modify a Db2 resource. This could be a plan, or a Db2 command, or a resource that is needed to execute dynamic SQL. At this stage, you can use Db2 security checking, which is managed either by Db2 itself, or by RACF or an equivalent external security manager, to control the CICS user's access to the resource.

You can also use RACF, or an equivalent external security manager, to protect the components that make up CICS and Db2 from unauthorized access. You can apply this protection to Db2 databases, logs, bootstrap data sets (BSDSs), and libraries outside the scope of Db2, and to CICS data sets and libraries. You can use VSAM password protection as a partial replacement for the protection provided by RACF. For more information, see CICS system resource security.

Note: RACF is referred to here as the external security manager used by CICS. Except for the explicit RACF examples, the general discussion applies equally to any functionally equivalent non-IBM external security manager.

Figure 23 on page 46 shows the security mechanisms involved in a CICS Db2 environment.

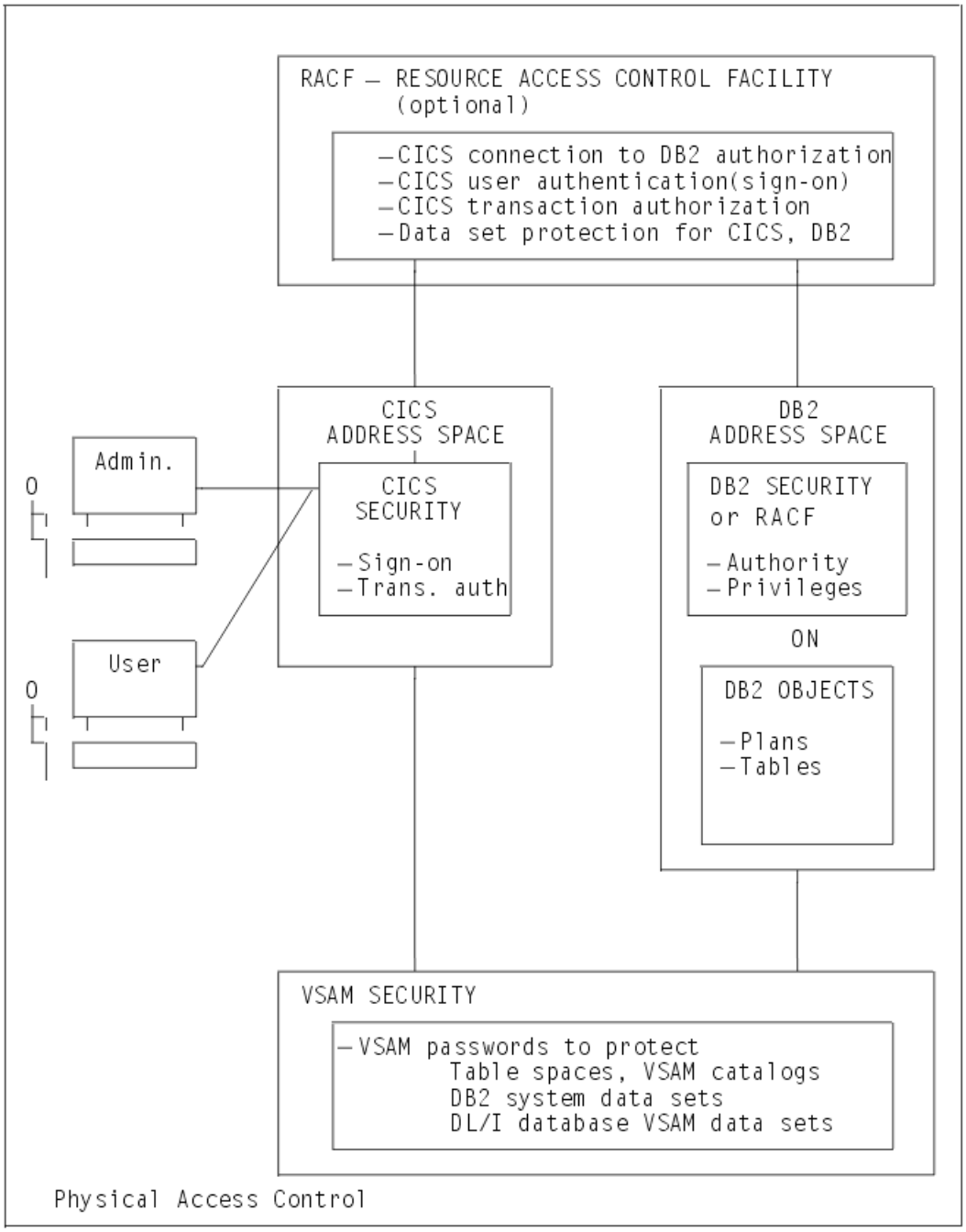


Figure 23. Overview of the CICS Db2 security mechanisms

Controlling access to Db2-related resources in CICS

You can control access to Db2 resources in your CICS region, and initiate security checking for the resources, by enabling an external security manager and the appropriate CICS security mechanism.

About this task

CICS users might want to perform the following activities involving Db2:

- Inquire on, modify, create or discard DB2CONN, DB2ENTRY, and DB2TRAN resource definitions.
- Use a transaction that accesses Db2 to obtain data, or issue CICS Db2 attachment facility commands or Db2 commands using the DSNC transaction.

Use RACF, or an equivalent external security manager to perform security checks in your CICS region. When a user tries to access protected resources, CICS calls the external security manager to perform security checking. RACF makes security checks using the CICS user's ID, which is authenticated when the user signs on to CICS. If a user does not sign on to CICS, they are given the default user ID, unless a user ID has been permanently associated with a terminal using preset security.

Enable the appropriate security mechanism for your CICS region: transaction-attach security, resource security, command security, or surrogate security.

For more information about CICS security, see CICS TS security.

Controlling users' access to DB2CONN, DB2TRAN, and DB2ENTRY resource definitions

You can control users' access to DB2CONN, DB2TRAN, and DB2ENTRY resource definitions by enabling different CICS security mechanisms.

About this task

The following security mechanisms can be used to control user access to Db2 resource definitions.

- Control users' ability to access particular resources by using the CICS **resource security** mechanism. Resource security is implemented at the transaction level. For example, you could prevent some users from modifying a particular DB2ENTRY definition. “Using resource security to control access to DB2ENTRY and DB2TRAN resource definitions” on page 47 tells you how to use this security mechanism.
- Control users' ability to issue particular SPI commands against Db2-related resources by using the CICS **command security** mechanism. Command security is also implemented at the transaction level. For example, you could permit only certain users to issue CREATE and DISCARD commands against DB2ENTRY resource definitions. “Using command security to control the issuing of SPI commands against DB2CONN, DB2ENTRY, and DB2TRAN resource definitions” on page 49 tells you how to use this security mechanism.
- Control users' ability to modify the authorization IDs that CICS provides to Db2, by using the CICS **surrogate security and AUTHTYPE security** mechanisms. The authorization IDs are used for Db2 security checking, and they are set by the AUTHID, COMAUTHID, AUTHTYPE and COMAUTHTYPE attributes on Db2-related resource definitions, and by the SIGNID attribute on the DB2CONN definition for the CICS region. CICS checks that the user who wants to modify the authorization ID, is permitted to act on behalf of the existing authorization ID that is specified in the resource definition. “Using surrogate security and AUTHTYPE security to control access to the authorization IDs that CICS provides to Db2” on page 50 tells you how to use these security mechanisms.

Using resource security to control access to DB2ENTRY and DB2TRAN resource definitions

The CICS resource security mechanism controls users' access to named CICS resources. For example, you can use it to protect certain resources, such as a particular DB2ENTRY definition, from being modified by particular users.

About this task

CICS command security can prevent users from performing certain actions on types of resources, such as "all DB2ENTRY definitions", but it cannot protect individual items within the resource type.

Because your CICS region only has one DB2CONN definition, you do not need to use resource security to protect it; you can control access to the DB2CONN definition using command security. Also, DB2TRAN definitions, for the purpose of resource security, are treated as extensions of the DB2ENTRY definition to which they refer, and are not defined for resource security in their own right. If you give a user permission to access a DB2ENTRY definition, you also give them permission to access the DB2TRAN definitions that refer to it. (In the case where a transaction changes the name of the DB2ENTRY with which a DB2TRAN

definition is associated, a double security check is performed, to verify the user's authority to modify both the old DB2ENTRY to which the definition referred, and the new DB2ENTRY to which it will refer.) For resource security, you therefore only need to define your DB2ENTRY definitions to RACF.

When resource security is enabled for a transaction, the external security manager checks that the user ID associated with the transaction is authorized to modify the resource that is involved. Security of resource definitions has more information about this process.

To protect your Db2-related resources using resource security, complete these steps.

Procedure

1. To enable RACF, or an equivalent external security manager, and make resource security available for a CICS region, specify SEC=YES as a system initialization parameter for the CICS region.
2. In RACF, create general resource classes to contain your Db2-related resources. You need a member class and a grouping class.

   Unlike the RACF default resource class names for CICS, there are no IBM-supplied default class names for DB2ENTRY resources. Create your own installation-defined class names 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 CICSTS56.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. Do not use existing CICS class names for Db2-related resource definitions; instead, create new class names using a similar naming convention.
3. Define profiles for your DB2ENTRY definitions in the resource classes that you have created.

   For example, to add a number of DB2ENTRY names to the XCICSDB2 resource class, use the RDEFINE command as follows:

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

   Protecting DB2ENTRY resource definitions also protects access to associated DB2TRAN definitions, because a DB2TRAN is considered to be an extension to the DB2ENTRY to which it refers. You do not need to protect your DB2CONN definition using resource security.
4. To activate resource security for your Db2-related resources, specify XDB2=*name* as a system initialization parameter for the CICS region, where *name* is the general resource class name that you defined for your Db2-related resources.
5. Specify RESSEC=YES in the resource definition for any transactions involving Db2-related resources for which you want to enable resource security. Now, when a user tries to use one of these transactions to access one of the Db2-related resources that you have protected, RACF checks that the user ID is authorized to access that resource.
6. Give permission to your CICS users, or groups of users, to perform appropriate actions on each Db2-related resource that you have protected.

   Remember that if a user has permission to perform actions on a DB2ENTRY definition, they are automatically authorized to perform the same actions on DB2TRAN definitions associated with it.The access that users need to perform certain actions is as follows:

   INQUIRE command
   Requires READ authority

   SET command
   Requires UPDATE authority

   CREATE command
   Requires ALTER authority

   DISCARD command
   Requires ALTER authority

For example, you can use the PERMIT command to authorize a group of users to modify a protected DB2ENTRY, db2ent1 in class XCICSDB2, with UPDATE authority, as follows:

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

Using command security to control the issuing of SPI commands against DB2CONN, DB2ENTRY, and DB2TRAN resource definitions

Use CICS command security mechanisms to protect DB2CONN, DB2ENTRY, and DB2TRAN resource definitions.

About this task

The CICS command security mechanism controls users' ability to issue particular SPI commands against types of Db2-related resource. For example, you can use it to control which users are allowed to issue CREATE and DISCARD commands against DB2ENTRY resource definitions. Unlike resource security, CICS command security cannot protect individual named resources; it is designed to protect types of resource. You can use command security to protect DB2CONN, DB2ENTRY, and DB2TRAN resource definitions.

When command security is enabled for a transaction, the external security manager checks that the user ID associated with the transaction is authorized to use that command to modify the type of resource that is involved. CICS command security has more information about this process.

If you have both resource security and command security enabled for a particular transaction, RACF performs two security checks against the user ID. For example, if a transaction involves the user issuing a DISCARD command against DB2ENTRY definition db2ent1, RACF checks:

1. That the user ID is authorized to issue the DISCARD command (ALTER authority) against the DB2ENTRY resource type.
2. That the user ID is authorized to access the DB2ENTRY definition db2ent1 with ALTER authority.

To protect your Db2-related resources using command security, complete these steps.

Procedure

1. To enable RACF, or an equivalent external security manager, for a CICS region, specify SEC=YES as a system initialization parameter for the region.
2. Add the Db2 resource names DB2CONN, DB2ENTRY, and DB2TRAN as resource identifiers in one of the IBM-supplied RACF resource classes for CICS commands, CCICSCMD or VCICSCMD.

   Alternatively, you can use a user-defined general resource class for your CICS commands. CICS resources subject to command security checking tells you more about this.

   For example, you can use the REDEFINE command to define a profile named CMDSAMP in the default class VCICSCMD, and use the ADDMEM operand to specify that the Db2 resource types are to be protected by this profile, as follows:

   ```
   RDEFINE  VCICSCMD CMDSAMP UACC(NONE)
                     NOTIFY(sys_admin_userid)
                     ADDMEM(DB2CONN, DB2ENTRY, DB2TRAN)
   ```

3. To make command security available for a CICS region:
   a) If you have used the IBM-supplied RACF resource classes CCICSCMD or VCICSCMD for CICS command profiles, specify XCMD=YES as a system initialization parameter for the region.

      Specifying YES means that CCICSCMD and VCICSCMD are used to build RACF's in-storage profiles.

   b) If you have used a user-defined general resource class for CICS commands, specify XCMD=*user_class* as a system initialization parameter for the region, where *user_class* is the name of the user-defined general resource class.
4. Specify CMDSEC=YES in the resource definition for any transactions involving Db2-related resources for which you want to enable command security. Now, when a user tries to use one of these

transactions to issue a command to modify one of the Db2-related resources that you have protected, RACF checks that the user ID is authorized to issue that command against that type of resource.

5. Give permission to your CICS users, or groups of users, to issue appropriate commands against each type of Db2-related resource. For command security, you need to give separate permissions relating to the DB2TRAN resource type, as well as to the DB2ENTRY resource type. You can also protect the DB2CONN resource type (that is, the CICS region's DB2CONN definition).

   The access that users need to issue certain commands is as follows:

   INQUIRE command
   Requires READ authority

   SET command
   Requires UPDATE authority

   CREATE command
   Requires ALTER authority

   DISCARD command
   Requires ALTER authority

   For example, if you have defined the Db2 resource types in the CMDSAMP profile as in the example in Step 2, you can use the PERMIT command to authorize a group of users to issue EXEC CICS INQUIRE commands against the Db2 resource types as follows:

```
PERMIT CMDSAMP CLASS(VCICSCMD) ID(operator_group) ACCESS(READ)
```

   Within a transaction, you can query whether a user ID has access to Db2 resource types by using the **EXEC CICS QUERY SECURITY RESTYPE(SPCOMMAND)** command, with the RESID parameter specifying DB2CONN, DB2ENTRY, or DB2TRAN.

Using surrogate security and AUTHTYPE security to control access to the authorization IDs that CICS provides to Db2

The CICS surrogate security and AUTHTYPE security mechanisms control users' ability to modify the authorization IDs that CICS provides to Db2.

About this task

Use surrogate security and AUTHTYPE security to ensure that only certain users are permitted to change the authorization IDs, which are used for security checking by Db2 own security checking. Surrogate security and AUTHTYPE security are set for the whole CICS region, and any transactions that involve changes to the authorization IDs are subject to them.

“Providing authorization IDs to Db2 for the CICS region and for CICS transactions” on page 53 explains how to select and change these authorization IDs. To summarize, the authorization IDs that CICS provides to Db2 are set by the AUTHID, COMAUTHID, AUTHTYPE, and COMAUTHTYPE attributes on Db2-related resource definitions, and by the SIGNID attribute on the DB2CONN definition for the CICS region. To change the authorization IDs, you first need authority to modify the DB2CONN and DB2ENTRY definitions, which might be protected by command security or resource security. Surrogate security provides an extra layer of protection, because it involves CICS acting on behalf of Db2 to check that the user that is modifying the authorization ID is permitted to act as a surrogate for the existing authorization ID that is specified in the resource definition.

True surrogate security provides security checking when a user attempts to change the SIGNID, AUTHID or COMAUTHID attributes on a DB2CONN or DB2ENTRY definition, all of which specify an authorization ID that is used when a process signs on to Db2. CICS uses the surrogate user facility of RACF to perform this checking. A surrogate user is one who has the authority to do work on behalf of another user, without knowing that other user's password. When a user attempts to change one of the SIGNID, AUTHID or COMAUTHID attributes, CICS calls RACF to check that the user is authorized as a surrogate of the authorization ID that is presently specified on the SIGNID, AUTHID or COMAUTHID attribute.

For the AUTHTYPE and COMAUTHTYPE attributes, which give a type of authorization ID to be used rather than specifying an exact authorization ID, CICS cannot use true surrogate security. Instead, it uses a mechanism called AUTHTYPE security. When a user attempts to change one of the AUTHTYPE or COMAUTHTYPE attributes, CICS calls RACF to check that the user is authorized through a profile that you have defined for the resource definition in the RACF FACILITY general resource class. Although AUTHTYPE security is not true surrogate security, it is enabled by the same system initialization parameter, and you will probably want to use it in addition to surrogate security, so the instructions in this topic tell you how to set up both types of security.

Note that when DB2CONN and DB2ENTRY resource definitions are installed as part of a cold or initial start of CICS, if surrogate security and AUTHTYPE security are enabled, RACF makes surrogate security and AUTHTYPE security checks for the CICS region user ID. If you install DB2CONN and DB2ENTRY resource definitions in this way, ensure that the CICS region user ID is defined as a surrogate user for any authorization IDs specified in the resource definitions, and also that it is authorized through the correct profiles in the RACF FACILITY general resource class.

To implement surrogate security and AUTHTYPE security to protect the authorization IDs that CICS provides to Db2, complete the following steps:

Procedure

1. To enable RACF, or an equivalent external security manager, for a CICS region, specify SEC=YES as a system initialization parameter for the region.
2. To activate surrogate security and AUTHTYPE security for a CICS region, specify XUSER=YES as a system initialization parameter for the region. This system initialization parameter enables both the security mechanisms. When the security mechanisms are enabled, CICS calls RACF to perform security checks whenever a transaction involves EXEC CICS SET, CREATE, and INSTALL commands that operate on the SIGNID, AUTHID, COMAUTHID, AUTHTYPE and COMAUTHTYPE attributes of DB2CONN and DB2ENTRY resource definitions. For the SIGNID, AUTHID and COMAUTHID attributes, RACF performs the surrogate security check, and for the AUTHTYPE or COMAUTHTYPE attributes, RACF performs the AUTHTYPE security check.
3. For the purpose of surrogate security, you need to define appropriate CICS users, or groups of users, as surrogates of any authorization IDs that are specified on the SIGNID, AUTHID or COMAUTHID attributes of your DB2CONN and DB2ENTRY definitions. To define a user ID as a surrogate of an authorization ID:
   a) Create a profile for the authorization ID in the RACF SURROGAT class, with a name of the form *authid*.DFHINSTL, with the authorization ID defined as the owner.
   For example, if you have specified DB2AUTH1 as an authorization ID on a SIGNID, AUTHID or COMAUTHID attribute, use the following command to create a profile:

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

   b) Permit appropriate CICS users to act as a surrogate for the authorization ID, by giving them READ authority to the profile you have created.
   For example, to permit a user with the ID CICSUSR1 to act as a surrogate for the authorization ID DB2AUTH1, and therefore to install or modify any SIGNID, AUTHID or COMAUTHID attributes that specify DB2AUTH1 as the existing authorization ID, use the following command:

   ```
   PERMIT   DB2AUTH1.DFHINSTL  CLASS(SURROGAT) ID(CICSUSR1)  ACCESS(READ)
   ```

   Repeat this process for all the authorization IDs that you have specified on SIGNID, AUTHID or COMAUTHID attributes.

   c) If you might need to install DB2CONN and DB2ENTRY resource definitions containing SIGNID, AUTHID or COMAUTHID attributes as part of a cold or initial start of CICS, permit the CICS region user ID to act as a surrogate for any authorization IDs specified by those attributes.

   The defaults for DB2CONN and DB2ENTRY resource definitions do not involve the AUTHID and COMAUTHID attributes. The default SIGNID for an installed DB2CONN definition is the applid of the CICS region.

4. For the purpose of AUTHTYPE security, you need to create a profile for each of your DB2CONN or DB2ENTRY resource definitions in the RACF FACILITY general resource class, and give appropriate CICS users, or groups of users, READ authority to the profiles. (This process imitates the true surrogate security mechanism, but does not involve the use of a specific authorization ID; instead, it protects each resource definition.) To do this:

   a) Create a profile for the DB2CONN or DB2ENTRY resource definition in the RACF FACILITY general resource class, with a name of the form DFHDB2.AUTHTYPE.*authname*, where *authname* is the name of the DB2CONN or DB2ENTRY resource definition.
   For example, to define a profile for a DB2CONN resource definition named DB2CONN1, use the following command:

   ```
   RDEFINE  FACILITY  DFHDB2.AUTHTYPE.DB2CONN1   UACC(NONE)
   ```

   b) Give appropriate CICS users READ authority to the profile you have created.
   For example, to permit a user with the ID CICSUSR2 to install or modify the AUTHTYPE or COMAUTHTYPE attributes on a DB2CONN resource definition named DB2CONN1, use the following command:

   ```
   PERMIT  DFHDB2.AUTHTYPE.DB2CONN1 CLASS(FACILITY) ID(CICSUSR2) ACCESS(READ)
   ```

   Repeat this process for each of your DB2CONN and DB2ENTRY resource definitions. Also, if you might need to install DB2CONN and DB2ENTRY resource definitions containing AUTHTYPE or COMAUTHTYPE attributes as part of a cold or initial start of CICS, give READ authority to the CICS region user ID on the profiles for those resource definitions.

Controlling users' access to Db2-related CICS transactions

Use the CICS transaction-attach security mechanism to control users' access to: CICS transactions that access Db2 to obtain data, the DSNC transaction, and to any other transactions that issue CICS Db2 attachment facility commands and Db2 commands.

About this task

When transaction-attach security is enabled, RACF, or an equivalent external security manager, checks that the CICS user is authorized to run the transaction that they have requested.

To protect Db2-related transactions using transaction-attach security, follow the instructions in Transaction security. The process is the same for all CICS transactions; there are no special considerations for Db2-related transactions as far as the transaction-attach security mechanism is concerned. The instructions tell you how to:

- Set up appropriate system initialization parameters for the CICS region to activate transaction-attach security (see CICS parameters controlling transaction-attach security).
- Define transaction profiles to RACF for the transactions that you want to protect (see RACF profiles).

If you have defined transactions other than DSNC to issue CICS Db2 attachment facility commands and Db2 commands (for example, if you have created a separate transaction to run each command), remember to define these transactions to RACF as well.

You can now control which CICS users can use transactions that access Db2. Add the appropriate users or groups of users to the access list for the transaction profiles, with READ authority. RACF profiles has some recommendations about this.

For transactions that issue CICS Db2 attachment facility commands and Db2 commands, bear in mind that:

- CICS Db2 attachment facility commands operate on the connection between CICS and Db2, and they run entirely within CICS. Db2 commands operate in Db2 itself, and they are routed to Db2. You can distinguish Db2 commands from CICS Db2 attachment facility commands by the hyphen (**-**) character, which is entered with Db2 commands.

- If you have access to the DSNC transaction, CICS allows you to issue all of the CICS Db2 attachment facility commands and Db2 commands.
- If you have defined separate transactions to run individual CICS Db2 attachment facility commands and Db2 commands, you can give different CICS users authorization to subsets of these transaction codes, and therefore to subsets of the commands. For example, you could give some users authority to issue CICS Db2 attachment facility commands, but not Db2 commands. CICS-supplied transactions for CICS Db2 has the names of the separate transaction definitions that CICS supplies for the CICS Db2 attachment facility commands and the Db2 commands.

CICS Db2 attachment facility commands do not flow to Db2, so they are not subject to any further security checking. They are only protected by CICS transaction-attach security. However, Db2 commands, and CICS transactions that access Db2 to obtain data, are subject to further stages of security checking by the Db2 security mechanisms, as follows:

- When a transaction signs on to Db2, it must provide valid authorization IDs to Db2. The authorization IDs are checked by RACF or an equivalent external security manager.
- Because the transaction is issuing a Db2 command or accessing Db2 data, the authorization IDs that it has provided must have permission to perform these actions within Db2. In Db2, you can use GRANT statements to give the authorization IDs permission to perform actions.

In addition, the CICS region itself must be authorized to connect to the Db2 subsystem.

"Providing authorization IDs to Db2 for the CICS region and for CICS transactions" on page 53 tells you how to authorize the CICS region to connect to the Db2 subsystem, and how to provide valid authorization IDs for transactions.

"Authorizing users to access resources in Db2" on page 60 tells you how to grant permissions to the authorization IDs that the transactions have provided to Db2.

Providing authorization IDs to Db2 for the CICS region and for CICS transactions

CICS has two types of process that must provide Db2 with authorization IDs: the overall connection between a CICS region and Db2, and CICS transactions that acquire a thread into Db2.

About this task

For the purposes of security, Db2 uses the term "process" to represent all forms of access to data, either by users interacting directly with Db2, or by users interacting with Db2 by way of other programs, including CICS. A process that connects to or signs on to Db2 must provide one or more Db2 short identifiers, called authorization IDs, that can be used for security checking in the Db2 address space. Every process must provide a primary authorization ID, and it can optionally provide one or more secondary authorization IDs. Db2 privileges and authority can be granted to either primary or 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 that provides Db2 with the same secondary authorization ID has associated privileges over the table. To take privileges away from a user, the administrator can disconnect the user from that authorization ID.

CICS has two types of process that need to provide Db2 with authorization IDs:

- The overall connection between a CICS region and Db2, which is created by the CICS Db2 attachment facility. This process has to go through Db2 connection processing to provide Db2 with authorization IDs.
- CICS transactions that acquire a thread into Db2. These could be, for example, a transaction that is retrieving data from a Db2 database, or the DSNC transaction that is issuing a Db2 command. For each CICS transaction, the actual process that Db2 sees is the thread TCB, which CICS uses to control a transaction's thread into Db2. These processes have to go through the Db2 sign-on processing to provide Db2 with authorization IDs.

During connection processing and sign-on processing, Db2 sets the primary and secondary authorization IDs for the process to use in the Db2 address space. By default, Db2 uses the authorization IDs that the process has provided. However, both connection processing and sign-on processing involve exit routines, and these exit routines allow you to influence the setting of the primary and secondary authorization IDs. Db2 has a default connection exit routine and a default sign-on exit routine. You can replace these with your own exit routines, and a sample connection exit routine and sign-on exit routine are supplied with Db2 to assist you with this.

Providing authorization IDs to Db2 for a CICS region

CICS provides a primary authorization ID and one or more secondary authorization IDs to Db2.

About this task

When the CICS Db2 attachment facility creates the overall connection between a CICS region and Db2, the process goes through the Db2 connection processing. The CICS region can provide:

- A primary authorization ID. The primary authorization ID becomes the CICS region's primary ID in Db2. For the connection between a CICS region and Db2, you cannot choose the primary authorization ID that is initially passed to the Db2 connection processing; it is the user ID for the CICS region. However, it is possible to change the primary ID that Db2 sets during connection processing, by writing your own connection exit routine. If RACF, or an equivalent external security manager, is active, the user ID for the CICS region must be defined to it. “Providing a primary authorization ID for a CICS region” on page 54 tells you about the possible primary authorization IDs for a CICS region.
- One or more secondary authorization IDs. You can use the name of a RACF group, or list of groups, as secondary authorization IDs for the CICS region. If you do this, you need to replace the default Db2 connection exit routine DSN3@ATH, which only passes primary authorization IDs to Db2. The sample Db2 connection exit routine DSN3SATH passes the names of RACF groups to Db2 as secondary authorization IDs. Alternatively, you can write your own connection exit routine that sets secondary IDs for the CICS region. “Providing secondary authorization IDs for a CICS region” on page 55 tells you how to set up secondary authorization IDs for a CICS region.

Providing a primary authorization ID for a CICS region

The primary authorization ID that is passed to Db2 depends whether CICS is running as a started task, a started job, or a job.

About this task

The connection type that CICS requests from Db2 is single address space subsystem (SASS). For the connection between a CICS region and Db2, you cannot choose the primary authorization ID that is initially passed to the Db2 connection processing. The ID that is passed to Db2 as the primary authorization ID for the CICS region 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.

The user ID that a CICS region might use must be defined to RACF, or your equivalent external security manager, if the external security manager is active. Define the user ID to RACF as a USER profile. It is not sufficient to define it as a RESOURCE profile.

Once you have defined the CICS region's user ID to RACF, permit it to access Db2, as follows:

1. Define a profile for the Db2 subsystem with the single address space subsystem (SASS) type of connection, 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 the user ID for the CICS region to access the Db2 subsystem. For example, the following RACF command permits a CICS region with a user ID of CICSHA11 to connect to Db2 subsystem DB2A:

```
PERMIT DB2A.SASS CLASS(DSNR) ID(CICSHA11) ACCESS(READ)
```

The Db2 connection exit routine takes the primary authorization ID (the user ID) provided by the CICS region, and sets it as the primary ID for the CICS region in Db2. The default Db2 connection exit routine DSN3@ATH, and the sample Db2 connection exit routine DSN3SATH, both behave in this way. It is possible to change the primary ID that Db2 sets, by writing your own connection exit routine. For more information about the sample connection exit routine and about writing exit routines, see Securing Db2 in Db2 for z/OS product documentation. However, you might find it more straightforward to provide secondary authorization IDs for the CICS region, and grant permissions to the CICS region based on these, rather than on the primary authorization ID.

Providing secondary authorization IDs for a CICS region

When a CICS region connects to Db2, it can provide one or more secondary authorization IDs to Db2, in addition to the primary authorization ID. You can use the name of the RACF group, or list of groups, to which the CICS region is connected, as secondary authorization IDs. This enables you to grant Db2 privileges and authority to RACF groups, and then connect multiple CICS regions to the same groups, instead of granting Db2 privileges to all the possible primary authorization IDs for each CICS region.

About this task

To provide the name of the RACF group, or list of groups, to which a CICS region is connected, to Db2 as secondary authorization IDs, complete the following steps.

Procedure

1. Specify SEC=YES as a system initialization parameter for the CICS region, to ensure that CICS uses RACF.
2. Connect the CICS region to the appropriate RACF group or list of groups.

   See RACF group profiles.
3. Replace the default Db2 connection exit routine DSN3@ATH.

   This exit routine is driven during connection processing. The default connection exit routine does not support secondary authorization IDs, so you need to replace it with the sample connection exit routine DSN3SATH, which is provided with Db2, or with your own routine. DSN3SATH is shipped in source form in the Db2 SDSNSAMP library, and you can use it as a basis for your own routine. DSN3SATH passes the names of the RACF groups to which the CICS region is connected, to Db2 as secondary authorization IDs. If the RACF list of groups option is active, DSN3SATH obtains all the group names to which the CICS region is connected, and uses these as secondary authorization IDs. If the RACF list of groups option is not active, DSN3SATH uses the name of the CICS region's current connected group as the only secondary authorization ID.

Results

When the CICS region connects to Db2, the sample connection exit routine sets the CICS region's primary authorization ID (the region's user ID) as the primary ID, and sets the names of the RACF groups to which the CICS region is connected, as secondary IDs.

What to do next

As an alternative to providing the names of RACF groups as secondary authorization IDs to Db2, you can write your own connection exit routine that sets secondary IDs for the CICS region. For information about writing connection exit routines, see Securing Db2 in Db2 for z/OS product documentation.

Providing authorization IDs to Db2 for CICS transactions

So that CICS can pass authorization IDs to Db2, CICS must be using RACF; SEC=YES must be specified in the SIT. This is because CICS needs to pass a RACF access control environment element (ACEE) to Db2.

About this task

When a CICS transaction's thread TCB signs on to Db2 and goes through the Db2 sign-on processing, it can provide:

- A primary authorization ID. For CICS transactions, you can choose the primary authorization ID. It can be the user ID or operator ID of the CICS user, or a terminal ID, or a transaction ID; or it can be an ID that you have specified. The ID that is used as the primary authorization ID is determined by attributes in the DB2ENTRY definition (for entry threads), or in the DB2CONN definition (for pool threads and command threads). “Providing a primary authorization ID for CICS transactions” on page 57 tells you how to choose the primary authorization ID for a CICS transaction.
- One or more secondary authorization IDs. You can use the name of a RACF group, or list of groups, as secondary authorization IDs. This has the advantage that you can grant Db2 privileges and authority to RACF groups, rather than to each individual CICS user. To use secondary authorization IDs, use the AUTHTYPE attribute in the DB2ENTRY definition (for entry threads), or the AUTHTYPE or COMAUTHTYPE attributes in the DB2CONN definition (for pool threads or command threads), to specify the GROUP option. You also need to replace the default Db2 sign-on exit routine DSN3@SGN, because the default routine does not pass secondary authorization IDs to Db2. When you specify the GROUP option, the primary authorization ID is automatically defined as the user ID of the CICS user associated with the transaction. “Providing secondary authorization IDs for CICS transactions” on page 59 tells you how to set up and use secondary authorization IDs.

A key consideration for choosing the authorization IDs that CICS transactions provide to Db2 is the security mechanism that you have chosen for security checking in the Db2 address space. This security checking covers access to Db2 commands, plans, and dynamic SQL. You can choose to have this security checking carried out by:

- Db2 internal security.
- RACF, or an equivalent external security manager.
- Partly Db2, and partly RACF.

If you are using RACF for some or all of the security checking in your Db2 address space, CICS transactions that sign on to Db2 **must** provide an authorization ID by one of the following methods:

- Specify AUTHTYPE(USERID) or COMAUTHTYPE(USERID) in the appropriate definition for the thread (DB2ENTRY or DB2CONN), to provide the user ID of the CICS user associated with the transaction to Db2 as the primary authorization ID.
- Specify AUTHTYPE(GROUP) or COMAUTHTYPE(GROUP) in the appropriate definition for the thread (DB2ENTRY or DB2CONN), to provide the user ID of the CICS user associated with the transaction to Db2 as the primary authorization ID, and the name of a RACF group or list of groups as the secondary authorization IDs.
- Specify AUTHTYPE(SIGN) in the appropriate definition for the thread (DB2ENTRY or DB2CONN), and specify the CICS region user ID in the SIGNID attribute of the DB2CONN, to provide the CICS region ID to Db2 as the primary authorization ID.

Note that if the RACF access control environment element (ACEE) in the CICS region is changed in a way that affects the CICS Db2 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.

Providing a primary authorization ID for CICS transactions

When a CICS transaction's thread TCB signs on to Db2, it must provide a primary authorization ID to Db2. The ID that a transaction uses as its primary authorization ID is determined by an attribute in the resource definition for the thread that the transaction uses to access Db2.

About this task

This means that all transactions that use the same type of thread (either the same type of entry thread, or pool threads, or command threads) must use the same type of primary authorization ID. In each CICS region, you need to set a primary authorization ID for:

- Each type of entry thread, using your DB2ENTRY definitions.
- The pool threads, using your DB2CONN definition.
- The command threads (used for the DSNC transaction), using your DB2CONN definition.

Before you start to set primary authorization IDs, ensure that you have authority to do so. As well as having authority to change your DB2CONN or DB2ENTRY definitions, if surrogate user checking is in force for the CICS region (that is, the system initialization parameter XUSER is set to YES), you need to obtain special authority to perform operations involving Db2 authorization IDs. These operations are modifying the AUTHID, COMAUTHID, AUTHTYPE, or COMAUTHTYPE attributes on a DB2ENTRY or DB2CONN definition, and modifying the SIGNID attribute on a DB2CONN definition. “Using surrogate security and AUTHTYPE security to control access to the authorization IDs that CICS provides to Db2” on page 50 tells you how to grant users authority to perform these operations.

There are two methods of setting the primary authorization ID for a particular type of thread:

1. Use the AUTHID attribute in the DB2ENTRY definition (for entry threads), or the AUTHID or COMAUTHID attribute in the DB2CONN definition (for pool threads or command threads), to specify a primary authorization ID. For example, you could define `AUTHID=test2`. In this case, the CICS Db2 attachment facility passes the characters TEST2 to Db2 as the primary authorization ID.

   Using AUTHID or COMAUTHID does not permit the use of secondary authorization IDs, and also is not compatible with the use of RACF, or an equivalent external security manager, for security checking in the Db2 address space.
2. Use the AUTHTYPE attribute in the DB2ENTRY definition (for entry threads), or the AUTHTYPE or COMAUTHTYPE attribute in the DB2CONN definition (for pool threads or command threads), to instruct CICS to use an existing ID that is relevant to the transaction as the primary authorization ID. This ID can be a CICS user ID, operator ID, terminal ID, or transaction ID; or it can be an ID that you have specified in the DB2CONN definition for the CICS region.

   Using AUTHTYPE or COMAUTHTYPE is compatible with the use of RACF (or an equivalent external security manager) for security checking in the Db2 address space, if you use the USERID or GROUP options, and with the use of secondary authorization IDs, if you use the GROUP option.

The two methods of determining the primary authorization ID are mutually exclusive; you cannot specify both AUTHID and AUTHTYPE, or COMAUTHID and COMAUTHTYPE, in the same resource definition.

Remember that all IDs that you select as primary authorization IDs must be defined to RACF, or your equivalent external security manager, if the security manager is active for the Db2 subsystem. For RACF, the primary authorization IDs must be defined as RACF USER profiles, not just as RESOURCE profiles (for example, as a terminal or transaction).

Follow the instructions in DB2CONN resources and DB2ENTRY resources to set up or modify DB2CONN and DB2ENTRY definitions. If you are using the AUTHTYPE or COMAUTHTYPE attributes to determine the primary authorization ID for a type of thread, use Table 3 on page 58 to identify the options that provide the required authorization ID and support the facilities you want. The key points to consider are:

- If you want to provide secondary authorization IDs to Db2 as well as a primary authorization ID, you need to select the GROUP option. When you specify the GROUP option, your primary authorization ID is automatically defined as your CICS user ID, but you can base your security checking on the secondary authorization IDs instead.
- If you are using RACF for security checking in the Db2 address space, you need to select either the GROUP option, or the USERID option. Only these options can pass the RACF access control environment element (ACEE) to Db2, which is required when RACF is used for security checking.
- Think about the performance and maintenance implications of your choice of authorization ID. Selecting authorization IDs for performance and maintenance outlines these. With the USERID, OPID, TERM, TX or GROUP options, sign-on processing occurs more frequently, and maintenance also takes more time, because you need to grant permissions to a greater number of authorization IDs. With the SIGN option, or using the AUTHID attribute instead of the AUTHTYPE attribute, sign-on processing is decreased, and maintenance is less complicated. However, using standard authorization IDs makes the Db2 security checking less granular.
- Think about the accounting implications of your choice of authorization ID. The authorization ID is used in each Db2 accounting record. From an accounting viewpoint, the most detailed information is obtained if using USERID, OPID, GROUP or TERM. However, depending on the ACCOUNTREC specification, it may not be possible to account at the individual user level in any case. For more information about accounting in a CICS Db2 environment, see Accounting and monitoring in a CICS Db2 environment in Monitoring.

Table 3 on page 58 shows the primary authorization IDs that the CICS Db2 attachment facility passes to Db2 when you select each option for the AUTHTYPE or COMAUTHTYPE attributes.

Table 3. Options available on the AUTHTYPE and COMAUTHTYPE attributes

Option	Primary authorization ID passed to Db2	Supports RACF checking for Db2?	Supports secondary auth IDs?
USERID	User ID associated with the CICS transaction, as defined to RACF and used in CICS sign-on	Yes	No
OPID	User's CICS operator ID, defined in the CICS segment of the RACF user profile	No	No
SIGN	An ID you specified in the SIGNID attribute of the DB2CONN definition for the CICS region. Defaults to the applid of the CICS region	Yes, when the SIGNID attribute of the DB2CONN resource matches the CICS region userid.	No
TERM	Terminal ID of the terminal associated with the transaction	No	No
TX	Transaction ID	No	No
GROUP	User's CICS RACF user ID used in CICS sign-on	Yes	Yes

If you are not planning to provide secondary authorization IDs for your CICS transactions, you do not need to replace the default Db2 sign-on exit routine DSN3@SGN. The default sign-on exit routine handles primary authorization IDs. However, the Db2 subsystem to which you are connecting might use a different sign-on exit routine for some other reason. If the Db2 subsystem uses the sample sign-on exit routine DSN3SSGN, you might need to make a change to DSN3SSGN, if **all** of the following conditions are true:

- You have chosen an AUTHID or AUTHTYPE option other than GROUP.
- RACF list of groups processing is active.
- You have transactions whose primary authorization ID is not defined to RACF.

If this is the case, see Securing Db2 in Db2 for z/OS product documentationfor the change you need to make to the sample sign-on exit routine.

Providing secondary authorization IDs for CICS transactions

When a thread TCB that belongs to a CICS transaction signs on to Db2, it can provide one or more secondary authorization IDs to Db2, in addition to the primary authorization ID.

About this task

You can provide Db2 with the name of a CICS user's RACF group, or list of groups, as secondary authorization IDs. This has the advantage that you can grant Db2 privileges and authority to RACF groups, rather than to each individual CICS user. CICS users can then be connected to, or removed from, RACF groups as required. RACF group profiles explains how users can be placed into RACF groups.

You can only provide secondary authorization IDs to Db2 for CICS transactions if you specify the GROUP option for the AUTHTYPE attribute in the DB2ENTRY definition (for entry threads), or the AUTHTYPE or COMAUTHTYPE attributes in the DB2CONN definition (for pool threads or command threads). If you specify any other option for AUTHTYPE or COMAUTHTYPE, the secondary authorization ID is set to blanks. When you specify the GROUP option, you cannot choose the primary authorization ID for the thread type; it is automatically defined as the user ID of the CICS user associated with the transaction. You should base your security checking on the secondary authorization IDs instead.

To provide the names of a user's RACF groups to Db2 as secondary authorization IDs, complete the following steps:

1. Specify SEC=YES as a system initialization parameter for the CICS region, to ensure that CICS uses RACF. If your CICS transaction profile names are defined with a prefix, also specify the system initialization parameter SECPRFX=YES or SECPRFX=*prefix*.
2. If the CICS region is using MRO:
   a. Ensure that each connected CICS region is also using RACF security (SEC=YES).
   b. Specify ATTACHSEC=IDENTIFY on the CONNECTION definition for the TOR, to ensure that sign-on information is propagated from the TOR to the AORs.
3. Replace the default Db2 sign-on exit routine DSN3@SGN. This sign-on exit routine is driven during sign-on processing. The default sign-on exit routine does not support secondary authorization IDs, so you need to replace it with the sample sign-on exit routine DSN3SSGN, which is provided with Db2, or with your own routine. DSN3SSGN is shipped in source form in the Db2 SDSNSAMP library, and you can use it as a basis for your own routine. For information about sign-on exits and about writing exit routines, see Securing Db2 in Db2 for z/OS product documentation. If the RACF list of groups option is not active, DSN3SSGN passes the name of the current connected group as the secondary authorization ID. If the RACF list of groups is active, DSN3SSGN obtains the names of all the groups to which the user is connected, and passes them to Db2 as secondary authorization IDs.
4. Use the AUTHTYPE attribute in the DB2ENTRY definition (for entry threads), or the AUTHTYPE or COMAUTHTYPE attribute in the DB2CONN definition (for pool threads or command threads), to specify the GROUP option as the authorization type for each type of thread for which you want to provide secondary authorization IDs. If surrogate user checking is in force for the CICS region (that is, the system initialization parameter XUSER is set to YES), you need to obtain special authority to perform operations involving Db2 authorization IDs. “Using surrogate security and AUTHTYPE security to control access to the authorization IDs that CICS provides to Db2” on page 50 tells you how to do this.

If you have successfully completed all the steps listed above, when a CICS transaction's thread TCB signs on to Db2 with the types of thread that you have defined with the GROUP option, the CICS user's user ID will be passed to Db2 as the primary authorization ID, and the user's RACF group or list of groups will be passed to Db2 as secondary authorization IDs. If you have not successfully completed all the steps, the CICS Db2 attachment facility will issue an authorization failed message.

As an alternative to providing the names of RACF groups as secondary authorization IDs to Db2, you can write your own sign-on exit routine that sets secondary IDs for CICS transactions. See Securing Db2 in Db2 for z/OS product documentation.

Authorizing users to access resources in Db2

Once the user has accessed the Db2 address space from a CICS transaction they might need permission to issue Db2 commands or execute a plan.

About this task

Access to the Db2 resources that a CICS user needs to issue Db2 commands or execute a plan is subject to security checking by the Db2 security mechanisms. You can choose to have this security checking carried out by:

- Db2 internal security.
- RACF, or an equivalent external security manager.
- Partly Db2, and partly RACF.

For more information about setting up RACF to perform security checking in the Db2 address space, see Securing Db2 in Db2 for z/OS product documentation.

If you are using RACF for some or all of the security checking in your Db2 address space, remember that CICS transactions that sign on to Db2 must provide an authorization ID. For more information, see “Providing authorization IDs to Db2 for CICS transactions” on page 56. CICS must also be using RACF (SEC=YES must be specified in the SIT). This is because when RACF is used for security checking in the Db2 address space, CICS needs to pass a RACF access control environment element (ACEE) to Db2. CICS can only produce an ACEE if it has RACF active, and only threads defined with the GROUP, SIGN or USERID option can pass the ACEE to Db2.

When the ACEE is passed to Db2, it is used by the Db2 exit DSNX@XAC, which determines whether RACF, or an equivalent non-IBM external security manager, or Db2 internal security is used for security checking. DSNX@XAC is driven when a transaction whose thread has signed on to Db2, issues API requests. You can modify DSNX@XAC. For more information, see Securing Db2 in Db2 for z/OS product documentation.

Db2, or the external security manager, performs security checking using the authorization IDs that the CICS transaction provided to Db2 when the thread that it was using signed on to Db2. The authorization IDs could be related to the individual CICS user (for example, the CICS user's user ID and the RACF groups to which the user is connected), or they could be related to the transaction (for example, the terminal ID or transaction ID), or they could be related to the whole CICS region. For more information, see “Providing authorization IDs to Db2 for the CICS region and for CICS transactions” on page 53.

Db2, or the external security manager, checks that you have given the authorization IDs permission to perform the relevant actions in Db2. You can give the authorization IDs this permission by using GRANT statements in Db2. For information about how to grant, and revoke, Db2 permissions for authorization IDs, see Securing Db2 in Db2 for z/OS product documentation.

Controlling users' access to Db2 commands

For CICS users, the first security checking related to Db2 commands is performed in the CICS address space, when the user tries to access a CICS transaction that issues Db2 commands. This could be DSNC, or a user-defined transaction that invokes DFHD2CM1 and runs an individual Db2 command.

About this task

“Controlling users' access to Db2-related CICS transactions” on page 52 describes how to control users' access to transactions that issues Db2 commands in the CICS address space.

When a user issues a Db2 command through a CICS transaction, they are also subject to Db2 security checking, which verifies that they are authorized to Db2 to issue the command. This security checking

uses the authorization IDs (primary or secondary) that the transaction has passed from CICS. “Providing authorization IDs to Db2 for the CICS region and for CICS transactions” on page 53 tells you how to choose these authorization IDs and provide them to Db2. For transactions that use DFHD2CM1 to issue Db2 commands, the authorization IDs are set by the COMAUTHID or COMAUTHTYPE attribute of the CICS region's DB2CONN definition. For other applications that issue Db2 commands, the authorization IDs are set by the AUTHID or AUTHTYPE attribute for the CICS region's resource definition for the type of thread used by the transaction (pool thread or entry thread). These attributes control the authorization ID, or type of authorization ID, that is passed to Db2 by a transaction that is using that type of thread.

Db2 commands are therefore subject to two security checks, one in the CICS address space and one in the Db2 address space. Figure 24 on page 61 illustrates the process.

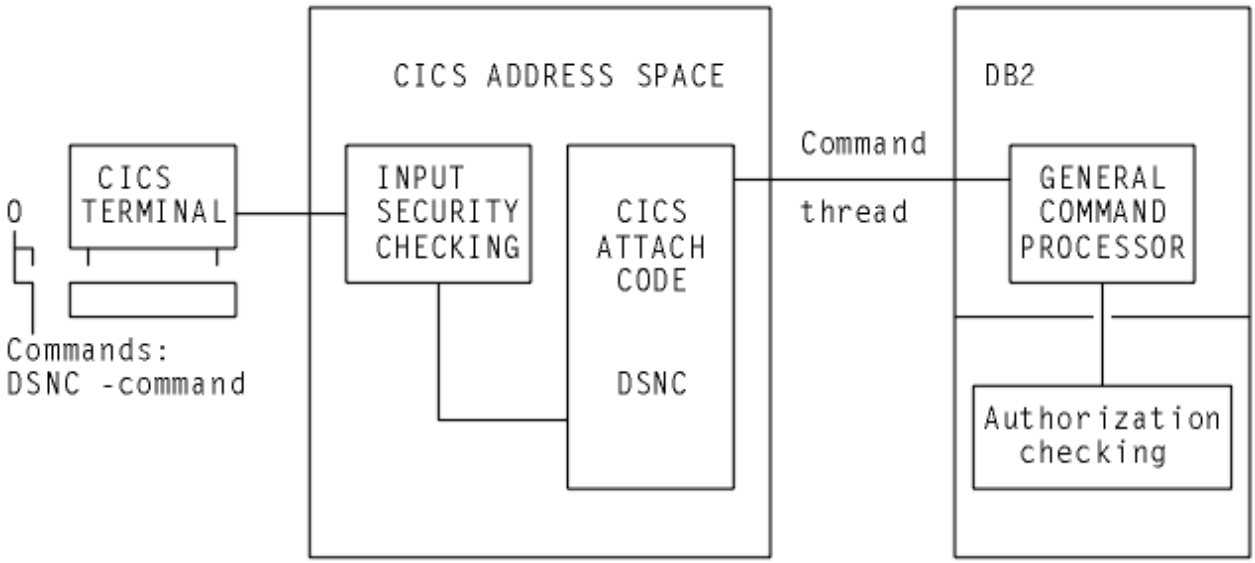


Figure 24. Security mechanisms for Db2 commands

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.

To authorize a user to issue a Db2 command, use a GRANT command to grant Db2 command privileges to the authorization ID that the transaction has passed from CICS. For information about how to grant, and revoke, Db2 permissions for authorization IDs, see Securing Db2 in Db2 for z/OS product documentation.

Controlling users' access to plans

A number of checks take place before a user can access plans in Db2. These checks start in the CICS address space before the request is passed to Db2 for further checks.

About this task

For Db2 commands, the first security check for users' access to plans takes place in the CICS address space, when CICS verifies that the user is permitted to access the transaction that will execute the plan. The second security check takes place in the Db2 address space, when Db2 verifies that the authorization ID provided by the transaction, is authorized to execute the plan. Figure 25 on page 62 illustrates this process.

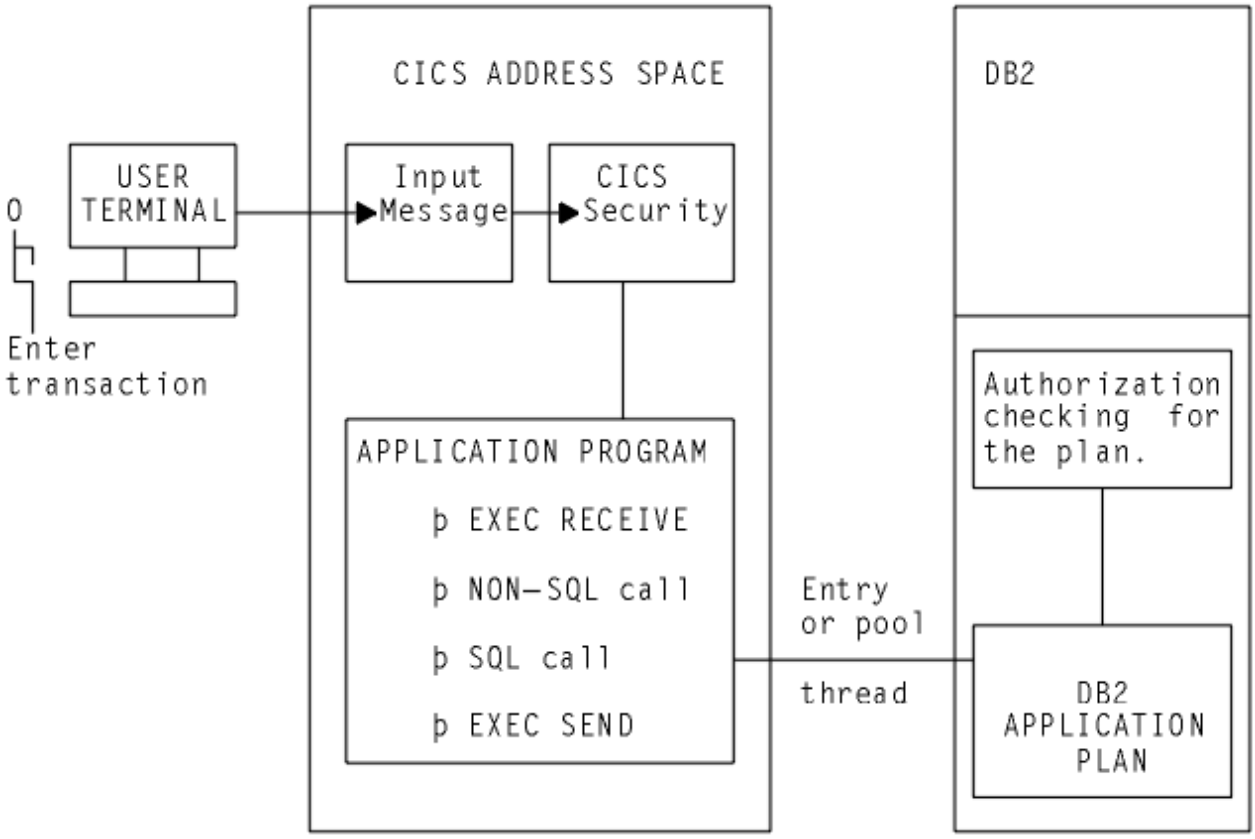


Figure 25. Security mechanisms for executing a plan

To authorize a user to execute a plan, use a GRANT command to grant Db2 command privileges to the authorization ID that the transaction has passed from CICS. For information about how to grant, and revoke, Db2 permissions for authorization IDs, see Securing Db2 in Db2 for z/OS product documentation.

If a plan includes dynamic 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.

About this task

However, if a plan includes the use of dynamic SQL, the authorization ID passed from CICS to Db2 must possess the privileges required to access all the Db2 resources involved, both the plan and the data. For example, if you 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.

If you do not want to spend too much time granting Db2 privileges, where a transaction executes a plan that involves the use of dynamic SQL, consider using one of the following methods of supplying an authorization ID to Db2:

- Use the SIGN option on the AUTHTYPE attribute of the DB2ENTRY definition for the thread used by the transaction. This results in the transaction having the primary authorization ID that you specified in the SIGNID attribute of the DB2CONN definition for the CICS region. (This method is not suitable where RACF is used for security checking in the Db2 address space.)
- Use the AUTHID attribute of the DB2ENTRY definition for the thread used by the transaction, to specify a standard authorization ID. Use the same authorization ID for all the transactions that need to access dynamic SQL. (This method is not suitable where RACF is used for security checking in the Db2 address space.)

- Create a RACF group, and connect your CICS users to this RACF group. Use the GROUP attribute of the DB2ENTRY definition for the thread used by the transaction, so that the RACF group is one of the secondary IDs that is passed to Db2.

In each case, you can then grant Db2 privileges for Db2 resources involved in all dynamic SQL to a single ID, either the standard authorization ID from the DB2CONN definition or the AUTHID attribute, or the name of the RACF group. "Providing authorization IDs to Db2 for the CICS region and for CICS transactions" on page 53 tells you how to provide authorization IDs by all these methods.

Db2 multilevel security and row-level security

DB2 Version 8 introduced support for multilevel security. CICS does not provide specific support for multilevel security, but you can use CICS in a multilevel-secure environment provided that you take care with the configuration.

For more information about multilevel security, see the following publications:

- Securing Db2 in Db2 for z/OS product documentation
- z/OS Planning for Multilevel Security and the Common Criteria
- IBM Redbooks: Securing DB2 and Implementing MLS on z/OS

When multilevel security is implemented at row level (row-level security) for data in DB2 Version 8 or later, the RACF SECLABEL class is activated, and a set of security labels is defined for users and for the Db2 table rows. The RACF options SETR MLS and MLACTIVE are not required to be active. You can use Db2 row-level security without impact on the rest of the MVS system.

CICS is able to access Db2 rows secured in this way. For CICS, you need to ensure that the RACF user profile for a CICS user that needs access to the Db2 rows is defined in RACF to include a default SECLABEL. For details, see the z/OS Security Server RACF Security Administrator's Guide.

When a CICS user signs on to a CICS region with SEC=YES specified in the SIT, RACF associates the default SECLABEL with the RACF access control environment element (ACEE) for the user. The DB2ENTRY definition (or DB2CONN definition if the pool is being used) needs to specify AUTHTYPE=USERID or AUTHTYPE=GROUP, which ensures the ACEE is passed on to Db2 for further security checking. An individual CICS user can therefore only have one associated SECLABEL.

For non-terminal tasks or programs, such as PLT programs, if the PLTPIUSR system initialization parameter is not specified and the PLTPISEC=NONE system initialization parameter is specified, PLT programs are run under the CICS region userid. In this case, you need to define the CICS region userid with a default SECLABEL. If you need to define different SECLABELS for a transaction, you would need to run each transaction in a separate CICS region which has a different CICS region userid and associated SECLABEL.

Chapter 5. Application design and development considerations for CICS Db2

During the design process

During the application design process, decisions you take can affect applications currently under development and planned future applications. Key design considerations include:

- The relationship between CICS applications and Db2 plans and packages. To gain the greatest benefits for the performance and administration of your system, the relationship between Db2 plans, transactions, and application programs must be defined when you are designing applications.
- The locking strategy. Locking is affected by options you choose when creating tables in Db2, by the design of your application programs, and by options you choose when binding plans, so you need to take it into account throughout the development process.
- Consider the security aspects of both the CICS Db2 test system and the CICS Db2 production system.
- Use of commands and programming techniques. These can improve the performance of your application and avoid potential problems. The use of qualified and unqualified SQL can also influence other aspects of the CICS Db2 environment. To run Java applications in the JVM server environment and gain the performance benefits of the open transaction environment (OTE), application programs must be threadsafe. For more information, see "SQL, threadsafe and other programming considerations for CICS Db2 applications" on page 80.
- Java programs in the CICS Db2 environment. For information about the support and programming considerations, see Chapter 6, "Using JDBC and SQLJ to access Db2 data from Java programs," on page 91 .
- Additional steps to take when application development is complete and the application is put into production.
- Connections defined for improved performance. Define your CICS Db2 connections to enhance application performance. For example, using protected entry threads for the application programs to access Db2 improves performance for heavily-used applications.

Factors affected by design

The design that you implement can influence:

- Performance
- Concurrency
- Operation
- Security
- Accounting
- The development environment

Factors that affect application performance

A well-designed CICS Db2 application should work properly when initially deployed into a production environment. However, some factors can affect the performance of well-designed applications:

- 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

Because of these factors, it is important to develop a consistent set of standards on the use of Db2 in the CICS environment.

Applications with data stored in VSAM and DL/1

For applications that have data stored in VSAM or DL/I, differences exist in the way in which some processes work, when compared with CICS Db2 applications. Differences to consider are:

- Locking mechanism
- Security
- Recovery and restart
- BIND process
- Operational procedures
- Performance
- Programming techniques

One of the main differences between batch program 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 related to performance.

Additional information

See the following additional sources of information on application design:

- Programming for Db2 for z/OS in Db2 for z/OS product documentation
- Designing applications

Designing the relationship between CICS applications and Db2 plans and packages

When CICS applications use Db2 data, the application architecture must take account of design aspects related to the CICS Db2 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 specified in the DB2CONN or DB2ENTRY definition for the thread that the transaction uses to access Db2. 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. The DBRMs can be bound directly into the plan, or they can be bound as packages and named in a package list in the plan.

To control the characteristics of the plan and the CICS Db2 attachment facility threads, the relationship between transaction IDs, Db2 plans, and the program modules must be defined in the design step. Some of the characteristics of the threads, environmental description manager (EDM) pool, and plans that depend on the design are listed below:

- 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

There are various design techniques for combining CICS transactions with Db2 plans, the recommended technique is to use plans based on packages.

A sample application

A simple example can be used to explain the consequences of different application design techniques.

Figure 26 on page 67 shows how CICS MAPs and transaction IDs are correlated, and how the transactions work, without Db2 considerations.

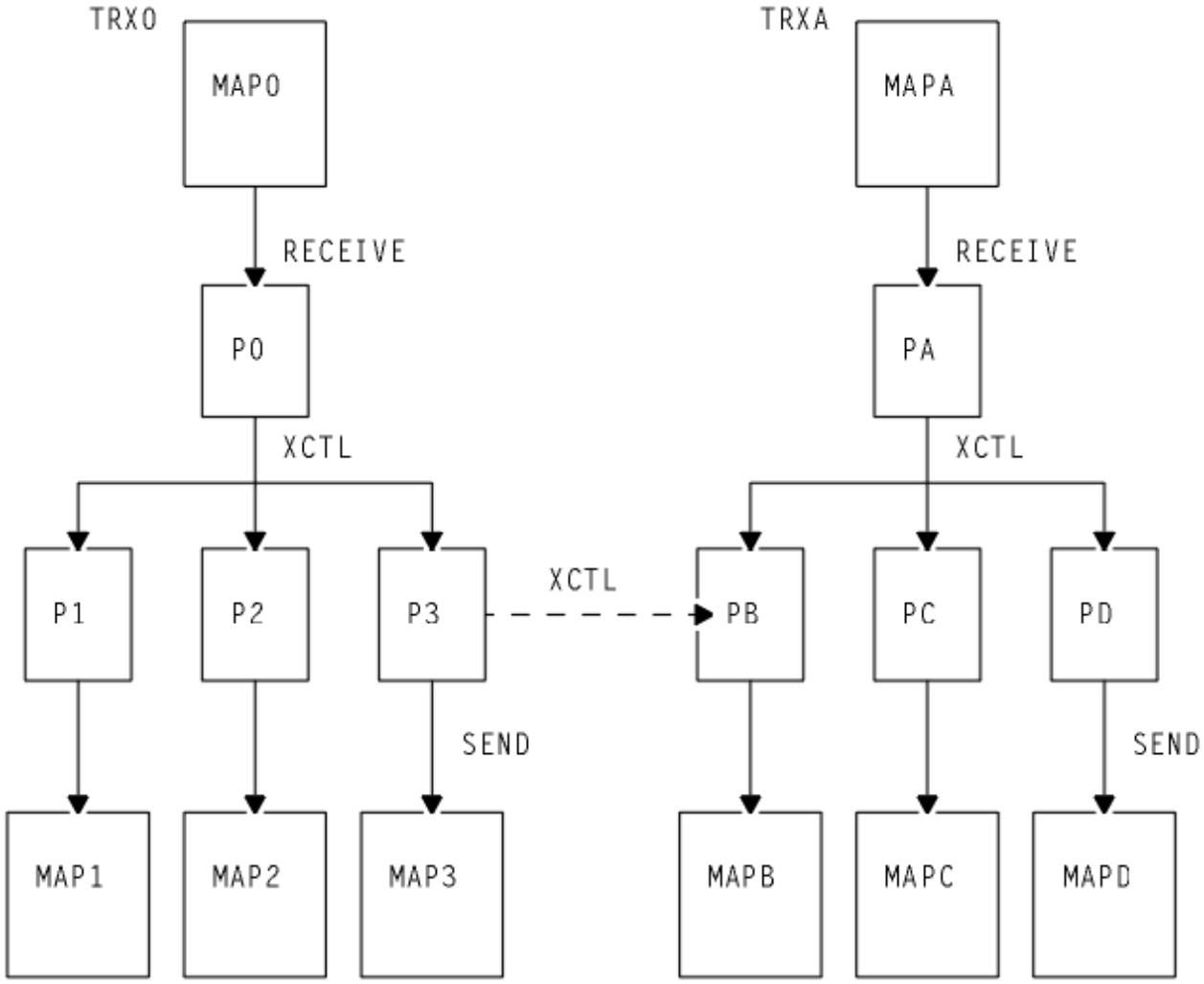


Figure 26. Example of a typical application design

In this example:

- The transaction ID, TRX0, is specified in the **EXEC CICS RETURN TRANSID(TRX0)** command, when a program (not shown) returns control after displaying MAP0.
- The next transaction then uses the transaction ID, TRX0, independent of what the terminal user decided to do.
- Program P0 is the initial program for transaction TRX0.
- We assume that all programs shown are issuing SQL calls.
- Depending on the option chosen by the terminal user, program P0 performs a CICS transfer control (XCTL) to one of the programs: P1, P2, or P3.
- After issuing some SQL calls, these programs display one of the maps: MAP1, MAP2, or MAP3.

- The example shown on the right side of the figure works in the same way.
- In some situations, program P3 transfers control to program PB.

Using Db2 packages

Using packages is the best way to ensure that the relationship between CICS transactions and Db2 plans is easy to manage.

In early releases of Db2, before packages were available, the DBRMs from all the programs that could run under a particular CICS transaction had to be directly bound into a single plan. Changing one DBRM in a plan (because the SQL calls for that program had changed) required all the DBRMs in the plan to be bound again. 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.

Dynamic plan exits were an interim solution designed to address this problem. The dynamic plan exit is an exit program that you specify in the DB2CONN or DB2ENTRY definition instead of specifying a plan name. You create many small plans for your CICS applications, each containing the DBRMs from a few programs, and the exit program selects the correct plan when each unit of work begins. You can also use the dynamic plan exit to switch between plans within a transaction. However, each small plan must still be rebound and taken out of commission every time an SQL statement changes in one of the programs that uses it. Also, the use of dynamic plan switching requires an implicit or explicit CICS **SYNCPOINT** to allow switching between plans.

Now that packages are available in Db2, using them is the best way to manage your plans. With packages, you can break the program units into much smaller parts, which you can rebind individually without affecting the entire plan, or even affecting the current users of the particular package you are rebinding.

Since updating a 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 definitions. You can also avoid situations where you might otherwise use dynamic SQL to obtain program flexibility. In a plan, you can even specify packages that do not exist yet, or specify package collections to which you can add packages. This means that DBRMs from new programs could be added without disturbing the existing plan at all.

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 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 Programming for Db2 for z/OS in Db2 for z/OS product documentation and the IBM Redbooks® publication, Db2 9 for z/OS : Packages Revisited .

Converting to packages

You can convert transactions that use dynamic plan exits or switching techniques to use packages instead.

About this task

A transaction that currently uses a dynamic plan exit or dynamic plan switching techniques can be converted to use packages as follows:

- Bind all of the DBRMs contained in the plan associated with the transaction into packages in 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 can now be used for this transaction to optimize thread reuse.

You could choose to have a single plan for the whole application, or one plan per transaction. The following sections give more detailed instructions for converting your transactions, based on this choice.

A similar approach can be taken for converting all CICS applications, whether they use a dynamic plan exit or not.

Note that high-usage packages can be bound with RELEASE(DEALLOCATE), with low-usage packages bound with RELEASE(COMMIT). This results in the high-usage packages being retained in the plan's package directory until plan deallocation, while the low-usage packages are removed from the directory, and the space in the EDM pool is released. The disadvantage of this approach is that if you use RELEASE(DEALLOCATE) and then need to rebind a highly-used package, you must intervene to force deallocation of the package if it is allocated to a long-running thread. Consider that to rebind a package is less expensive than to rebind a plan, in terms of the time spent doing it.

Using one plan for the application

Using one plan for the application gives the 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. Follow this procedure to convert to this environment.

Procedure

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 exit program name with the name of this new plan. For example, replace

   ```
   PLANEXITNAME=DSNCUEXT
   ```

   with

```
PLAN=PLANAPP1
```

Using one plan per transaction

This approach was recommended prior to DB2 Version 3 because accounting at the individual package level was not possible, which forced accounting to be done by plan name. However, current releases of Db2 provide accounting at the package level, which allows for plans to be made up of a large number of packages while still providing the required accounting granularity.

Procedure

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. 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:

     ```
     BIND PLAN (TRAN1) .... PKLIST (TRAN1.*) ...
     ```
   - Use `COLLAPP1.*` if a single collection was set up for all transactions:

     ```
     BIND PLAN (TRAN1) .... PKLIST (COLLAPP1.*) ...
     ```
3. Modify the DB2ENTRY definitions to replace the dynamic plan exit with the plan names associated with the transactions.

If you have packages and still want to use dynamic plan switching techniques

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 Db2 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. SQL: The language of Db2 in Db2 for z/OS product documentation lists the modifiable special registers.

Using one large plan for all transactions

A straightforward way to avoid having to switch plans during a transaction is to use one large plan for all CICS transactions that issue SQL calls.

For the example in Figure 26 on page 67:

- 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).
- 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 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.

Using many small plans

Using many small plans requires many transaction IDs, each of which has a corresponding plan. This technique reduces plan sizes and plan overlap.

You are recommended to use packages rather than using 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 26 on page 67, 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.

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.

Using plans based on transaction grouping

Transaction grouping can produce a number of midsize independent plans, where a plan switch can occur if required.

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 26 on page 67, 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. 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 produces 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.

Dynamic plan exits

You are recommended to use packages rather than dynamic plan exits, because of the advantages that packages provide. The use of dynamic plan exits was an interim solution that was designed to address problems in the CICS Db2 environment before packages were available 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).

Normally, 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. This is referred to as *dynamic plan switching*.

When a dynamic plan exit is used, 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 two sample assembler language exit programs, DSNCUEXT and DFHD2PXT, in both source and object code form. You can also write other exit programs.

Note that the use of the following items within the dynamic exit program is not supported:

- SQL commands.
- IFI calls.
- EXEC CICS SYNCPOINT commands.

The sample exit programs, DSNCUEXT and DFHD2PXT

Two sample assembler language exit programs, DSNCUEXT and DFHD2PXT, are supplied in both source and object code. The programs are functionally identical, but the CICS-supplied program definition for DSNCUEXT specifies CONCURRENCY(QUASIRENT), while the CICS-supplied program definition for DFHD2PXT specifies CONCURRENCY(THREADSAFE).

There are two sample programs because when CICS exploits the open transaction environment (OTE), the CICS Db2 task-related user exit operates as a threadsafe program and can receive control on an open TCB (L8 mode). For further explanation, see "Enabling CICS Db2 applications to use OTE through threadsafe programming" on page 81 . If the application program that made the Db2 request is threadsafe, it can also run on the open TCB. In this situation, TCB switching should not be needed.

However, if a dynamic plan exit is used that is defined with CONCURRENCY(QUASIRENT), like DSNCUEXT, this causes a switch back to the QR TCB, incurring an additional cost. Dynamic plan exits that are defined with CONCURRENCY(THREADSAFE) and are coded to threadsafe standards, like DFHD2PXT, can run on the open TCB, and do not incur the additional cost.

Therefore, when CICS is connected to Db2, use DFHD2PXT as your dynamic plan exit rather than DSNCUEXT. Remember that if you add logic that is not threadsafe to the supplied sample program, or issue non-threadsafe CICS commands in the exit, this will cause a switch back to the QR TCB, and the additional cost is incurred. To gain performance benefits by avoiding TCB switching, use a dynamic plan exit that is defined as threadsafe, code logic that is threadsafe, and ensure that the program contains only threadsafe commands.

The object code (load module) for the sample plan exits are included in the SDFHLOAD library. DSNCUEXT is the default dynamic plan exit, and it is invoked as a CICS user-replaceable program. The source code for both sample programs is written in assembler language and supplied in the SDFHSAMP library. The sample programs show how to address the parameter list, but do not change the plan name.

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

Table 4. Example of a parameter list passed to the DSNCUEXT or DFHD2PXT sample program through a COMMAREA

Assembler version format			
CPRMPLAN	DS	CL8	The DBRM/plan name of the first SQL statement on entry to the sample program. The field can be modified to establish a new plan.
CPRMAUTH	DS	CL8	The current authorization ID that is passed to Db2 at signon 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 the sample program. The CICS Db2 attachment preserves this field across invocations of the sample program.

Table 4. Example of a parameter list passed to the DSNCUEXT or DFHD2PXT sample program through a COMMAREA (continued)			
Assembler version format			
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.

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. As supplied by CICS , the dynamic plan exits DSNCUEXT and DFHD2PXT do not modify the plan name as input in CPRMPLAN by the CICS Db2 attachment facility, but return 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 Java applications for CICS , the CICS-supplied dynamic plan exits have been changed. SQLJ and JDBC require Db2 to produce four DBRMs for each application program to support dynamic change of isolation levels. For JDBC and SQLJ applications, the DBRM name is restricted to seven characters so that the eighth character can be used as a suffix of 1, 2, 3, or 4. Therefore, for JDBC and SQLJ applications you cannot use a default naming convention of *program name* equals *dbrm name* equals *plan name*, because the DBRM name has a suffix of 1, 2, 3 or 4.

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 always has the DBRM name set to the JDBC base program with the default isolation level appended; that is, by default, DSNJDBC2. For example, if the JDBC profile is generated using *pgmname=OTHER*, the DBRM name will be OTHER2. If dynamic plan exits are not used, the plan name is obtained from the DB2CONN or DB2ENTRY definition; the plan name in the properties file is ignored.

To support a default naming convention for the IBM Data Server Driver for JDBC and SQLJ, the CICS-supplied dynamic plan exits DSNCUEXT and DFHD2PXT can detect an input CPRMPLAN name whose first characters are SYSSTAT, SYSLH or SYSLN. If such a plan name is detected, the plan name is changed to "DSNJCC " (the seventh and eighth characters set to blanks). To use the default dynamic plan exits with Java applications for CICS, bind the multiple DBRMs into a plan called DSNJCC.

Writing your own exit program

You can write your exit program in assembler, COBOL, or PL/I.

About this task

The exit 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 Db2 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 Db2 attachment facility. The name specifies the plan name for this execution of the transaction.

If CICS is connected to DB2 Version 7 or later, then as explained in "Enabling CICS Db2 applications to use OTE through threadsafe programming" on page 81, you should ensure that the logic in your dynamic plan exit is coded to threadsafe standards, and that the program is defined as threadsafe (like the sample

exit program DFHD2PXT). If the program is not defined as threadsafe, each use of the program causes a switch back to the QR TCB, incurring an additional cost. If the program is defined as threadsafe but uses non-threadsafe CICS commands (which is permitted), each non-threadsafe command causes a switch back to the QR TCB and incurs the additional cost.

If you need to create plans for an application that has already been developed

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.

About this task

In general, defining plans after the application has already been developed is not recommended, but this technique is useful for *conversion projects*, where the application design is unchanged but the application now uses Db2.

The advantages and disadvantages of this technique completely depend on the actual application design and the result of the steps.

Procedure

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 on a plan that 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. The best way to manage the plans is to use packages, binding all the DBRMs into packages that are then listed in plans. For details, see “Using Db2 packages” on page 68.

   If you do not use packages, alternative techniques are described in “Using one large plan for all transactions” on page 70, “Using many small plans” on page 71, and “Using plans based on transaction grouping” on page 72.

Example

Applied to the example in Figure 26 on page 67, a possible solution would be:

- 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

If you need to switch plans within a transaction

The plan that a transaction uses to access Db2 must contain or reference the DBRMs from all the application programs that could operate under that transaction ID. If the plan doesn't contain all the relevant DBMRs you might consider switching plans within a transaction.

If the design of your applications has changed, some of your older plans might not contain all the relevant DBRMs. In this case, the application design could require a switch to a different plan in the course of a transaction, if the transaction requires that a program not included in the current plan must be executed. This situation could also occur if the structure of the Db2 plans was imperfectly considered during the design of the application.

The best solution to this problem is to use packages. You can bind the DBRM from the missing program into a package, and add the package to a package list in the existing plan. The transaction can now access the missing program without needing to switch plans. There are two other possible solutions, which are discussed as follows:

- "Dynamic plan switching" on page 76
- "Switching transaction IDs in order to switch plans" on page 76

Dynamic plan switching

Use dynamic plan switching (DPS) if you want to use more than one plan in a transaction.

Dynamic plan switching (DPS) allows you to use more than one plan in a transaction. However, as noted above, switching plans in 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. The dynamic plan exit can then be used to select the plan for the program you need to execute. This enables the use of a different plan for different UOWs within a transaction.

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.

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.

Switching transaction IDs in order to switch plans

Once a transaction has started, you cannot control the plan used by the transaction ID, but you can control the transaction ID itself. Therefore, you can switch plans by switching transaction IDs. However, you are not recommended to do this. Instead of switching transaction IDs to switch plans, you are recommended to bind the DBRM from the missing program into a package, and add the package to a package list in the existing plan.

About this task

If you have to switch transaction IDs, note that 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.

If you switch transaction IDs in order to switch plans, the application programs contain the logic to decide when to switch transaction ID. This means that if you want to change the plan structure (for example for performance and operational reasons), you need to change the application programs as well.

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.

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 to switching transaction IDs to switch plans 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 a control table to control program flow

You are recommended to use packages instead of this technique to control program flow.

The design principle presented 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.

Table 5 on page 77 shows an example of the contents of a control table. The example is based on the design situations described in Figure 26 on page 67.

Table 5. Control table for sample application

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	

Table 5. Control table for sample application (continued)				
Function name	**Program**	**Transaction**	**Plan name**	**New TRANS ID**
Price	PB	TEMP	PLANX	*

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

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.

The principle in the program flow is shown in Figure 27 on page 79.

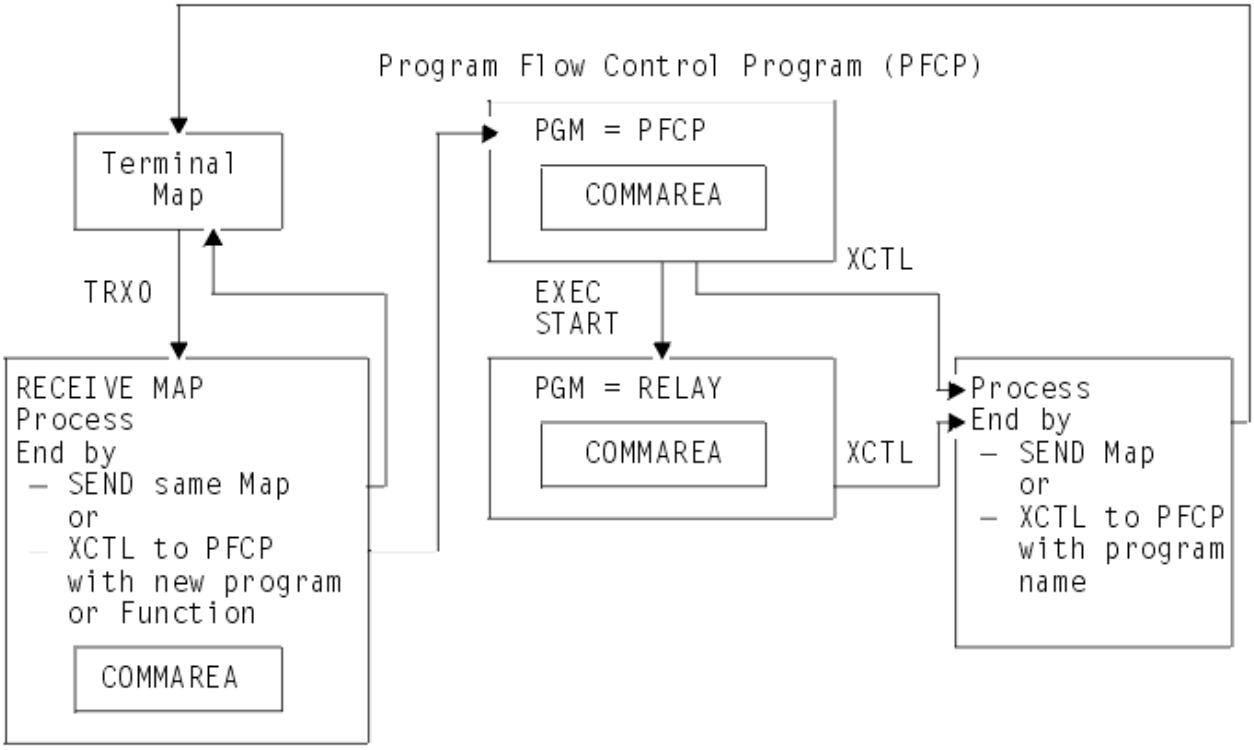


Figure 27. Table-controlled program flow

The flow for the application design used in Figure 26 on page 67 is as follows:

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 27 on page 79.

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.

Developing a locking strategy in the CICS Db2 environment

Db2 uses a lock mechanism to allow concurrency while maintaining data integrity.

In a CICS environment, concurrency is likely to be high. To give maximum concurrency, you should use row level locking or page locking instead of table space locking. You can do this by defining LOCKSIZE(PAGE), LOCKSIZE(ROW) or LOCKSIZE(ANY) when creating the table space, and by defining the isolation level as cursor stability at BIND time. For more information, see Managing Db2 performance in Db2 for z/OS product documentation.

Specifying LOCKSIZE(ANY) allows Db2 to decide if lock escalation can take place for the table space. The Db2 parameter NUMLKTS is the number of concurrent locks for a table space. If the number of locks exceeds NUMLKTS, lock escalation takes place. 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.

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.

You can override Db2 rules for choosing initial lock attributes by using the SQL statement LOCK TABLE in an application program. However, you should avoid using the LOCK TABLE statement, unless it is strictly necessary. If the LOCK TABLE statement is used in an online program, it can prevent the use of RELEASE(DEALLOCATE) and of protected threads. If you do use a LOCK TABLE statement, your plan should use the bind option RELEASE(COMMIT).

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.

SQL, threadsafe and other programming considerations for CICS Db2 applications

When developing application programs, you must be aware of certain programming considerations in order to improve the performance of your application and avoid potential problems. Whether you use qualified or unqualified SQL influences many other aspects of the CICS Db2 environment. To gain the

performance benefits of the open transaction environment (OTE), your application programs must be threadsafe.

Enabling CICS Db2 applications to use OTE through threadsafe programming

The CICS Db2 attachment facility includes a CICS Db2 task-related user exit (TRUE), DFHD2EX1, that is invoked when an application program makes an SQL request. It manages the process of acquiring a thread connection into Db2, and of returning control to the application program when the Db2 processing is complete.

About this task

The CICS Db2 attachment facility uses the OTE to enable the CICS Db2 TRUE to invoke and return from Db2 without switching TCBs. In the OTE, the CICS Db2 TRUE is automatically enabled using the REQUIRED option (but without the OPENAPI option, thereby meaning CICSAPI) on the ENABLE PROGRAM command during connection processing. This enables it to receive control on any type of key8 open TCB (an L8 or X8 or T8 TCB). Typically its an L8 TCB, but an X8 TCB is used for xplink C programs, and a T8 TCB used if its a Java program. Requests to Db2 are also issued on the open TCB, so it acts as the thread TCB, and no switch to a subtask TCB is needed.

In the OTE, if for example a cobol user application program that invoked the TRUE conforms to threadsafe coding conventions and is defined to CICS as threadsafe, it can also run on the L8 TCB. Before its first SQL request, the application program runs on the CICS main TCB, the QR TCB. When it makes an SQL request and invokes the TRUE, control passes to the L8 TCB, and Db2 processing is carried out. On return from Db2, if the application program is threadsafe, it now continues to run on the L8 TCB.

Programs defined with CONCURRENCY(REQUIRED) run on an open TCB from the start of the program. For CICSAPI programs, CICS uses an L8 open TCB regardless of the execution key of the program. For OPENAPI programs, CICS uses an L9 TCB if EXECKEY(USER) is set and an L8 TCB if EXECKEY(CICS) is set.

Where the correct conditions are met, the use of open TCBs for CICS Db2 applications decreases usage of the QR TCB, and avoids TCB switching. An ideal CICS Db2 application program for the OTE is a threadsafe program, containing only threadsafe EXEC CICS commands, and using only threadsafe user exit programs. An application like this moves to an L8 TCB when it makes its first SQL request, and then continues to run on the L8 TCB through any amount of Db2 requests and application code, requiring no TCB switching. This situation produces a significant performance improvement where an application program issues multiple SQL calls. If the application program does not issue many SQL calls, the performance benefits might not be as significant.

If the execution of the program involves any actions that are not threadsafe, CICS switches back to the QR TCB at that point. Such actions are non-threadsafe CICS requests issued by the program, the use of non-threadsafe dynamic plan exits, the use of non-threadsafe TRUEs, and the involvement of non-threadsafe global user exits (GLUEs). Switching back and forth between the open TCB and the QR TCB is detrimental to the performance of the application.

In order to gain the performance benefits of the OTE for CICS Db2 applications, you must meet the following conditions:

- CICS must be connected to a supported version of Db2. For details of the Db2 releases that are supported and the APARs that must be applied, see https://www.ibm.com/support/pages/uid/swg27018287.
- The system initialization parameter **FORCEQR** must not be set to YES. **FORCEQR** forces programs that are defined as threadsafe to run on the QR TCB, and it might be set to YES as a temporary measure while problems that are connected with threadsafe-defined programs are investigated and resolved.
- The CICS Db2 application must have threadsafe application logic (that is, the native language code in between the EXEC CICS commands must be threadsafe), use only threadsafe EXEC CICS commands, and be defined to CICS as threadsafe. Only code that has been identified as threadsafe is permitted to execute on open TCBs. If your CICS Db2 application is not defined as threadsafe, or if it uses EXEC CICS commands that are not threadsafe, TCB switching takes place and some or all of the performance benefits of OTE exploitation are lost.

- Any dynamic plan exits used by the CICS Db2 attachment facility must be coded to threadsafe standards and defined to CICS as threadsafe. The default dynamic plan exit DSNCUEXT, which is started as a CICS user-replaceable program, is not defined to CICS as threadsafe, but the alternative CICS-supplied sample dynamic plan exit DFHD2PXT is defined as threadsafe. See "Dynamic plan exits" on page 72 for more information.
- Any GLUEs on the execution path used by the application must be coded to threadsafe standards and defined to CICS as threadsafe (for CICS Db2 applications. In particular the GLUEs XRMIIN and XRMIOUT).
- Any other TRUEs used by the application must be defined to CICS as threadsafe, or as OPENAPI.

See Threadsafe programs for information about how to make application programs and user exit programs threadsafe. By defining a program to CICS as threadsafe, you are only specifying that the application logic is threadsafe, not that all the EXEC CICS commands included in the program are threadsafe. CICS can ensure that EXEC CICS commands are processed safely by switching to the QR TCB for those commands not yet converted that still rely on quasi-reentrancy. In order to permit your program to run on an open TCB, CICS needs you to guarantee that your application logic is threadsafe.

The EXEC CICS commands that are threadsafe, and so do not involve TCB switching, are indicated in the command syntax diagrams in the description of the API and SPI commands.

If a user application program in the OTE is not defined as threadsafe, the CICS Db2 TRUE still runs on an L8 TCB, but the application program runs on the QR TCB throughout the task. Every time the program makes an SQL request, CICS switches from the QR TCB to the L8 TCB and back again, so the performance benefits of the OTE are negated. The maximum TCB switching for a CICS Db2 application would occur if your program used a non-threadsafe user exit program and a non-threadsafe EXEC CICS command after every Db2 request. In particular, the use of a non-threadsafe exit program on the CICS-Db2 mainline path (for example, a program that is enabled at XRMIIN or XRMIOUT) causes more TCB switching than the level that is experienced when CICS is connected to earlier versions of Db2.

The table shows what happens when application programs with different concurrency attributes invoke the CICS Db2 TRUE. We shall assume it is not an xplink C program nor a Java program, so an L8 TCB is used.

Table 6. Combinations of application programs and the CICS Db2 TRUE

Program's concurrency attribute	**CICS Db2 TRUE's operation**	**Effect**
QUASIRENT	Runs on an L8 TCB	Application program runs under the CICS QR TCB. TRUEs run under an L8 TCB, and Db2 requests are executed under the L8 TCB. CICS switches to and from the CICS QR TCB and the L8 TCB for each Db2 request.
THREADSAFE	Runs on an L8 TCB	OTE exploitation. TRUEs run under an L8 TCB, and Db2 requests are executed under the L8 TCB. The application program also runs on the L8 TCB when control is returned to it. No TCB switches are needed until the task terminates, or if it issues a non-threadsafe CICS request which forces a switch back to the QR TCB.

Table 6. Combinations of application programs and the CICS Db2 TRUE (continued)		
Program's concurrency attribute	**CICS Db2 TRUE's operation**	**Effect**
REQUIRED with API(CICSAPI)	Runs on an L8 TCB	OTE exploitation. TRUEs run under an L8 TCB, and Db2 requests are executed under the L8 TCB. The application program runs on the L8 TCB from the start. The program always uses an L8 TCB irrespective of the execution key of the program. No TCB switches are needed until the task terminates, or if it issues a non-threadsafe CICS request which forces a switch back to the QR TCB and then a switch back afterward to the L8 TCB.
REQUIRED with API(OPENAPI)	Runs on an L8 TCB	OTE exploitation. Not preferred for user key CICS-Db2 applications (and when storage protection is active), as it causes switching from the L9 TCB to the L8 TCB and back again for every Db2 request.

In summary, to gain the performance benefits of the OTE:

- CICS must be connected to Db2.
- FORCEQR must not be set to YES.
- The CICS Db2 application must have threadsafe application logic (that is, the native language code in between the EXEC CICS commands must be threadsafe). If the application logic is not threadsafe, the program must be defined as CONCURRENCY(QUASIRENT), and so must operate on the CICS QR TCB. In this case TCB switching occurs for every Db2 request, even if the TRUE is running on an open TCB.
- A threadsafe application can be defined to CICS as CONCURRENCY(THREADSAFE) API(CICSAPI) or CONCURRENCY(REQUIRED) API(CICSAPI). The setting to use depends on how many non-threadsafe EXEC commands the program uses. If there are many non-threadsafe CICS commands the program is best defined as CONCURRENCY(THREADSAFE). If the program has few or no non-threadsafe CICS commands, then CONCURRENCY(REQUIRED) can be used. Programs defined with CONCURRENCY(REQUIRED) have the benefit of starting on an L8 open TCB, but every non-threadsafe CICS command results in two TCB switches.
- The CICS Db2 application must use only threadsafe or open API dynamic plan exits, TRUEs, and GLUEs. If any non-threadsafe exits are used, this forces a switch back to the QR TCB.

If all these conditions are met, you can gain the performance benefits of the OTE.

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 Programming for Db2 for z/OS in Db2 for z/OS product documentation.

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.

Using qualified and unqualified SQL

Programmers writing CICS Db2 programs can use qualified or 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 84.

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 on page 84 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.

It is recommended that you use qualified SQL for dynamic SQL statements, because it is easier to administer.

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.

Views

It is generally recommended that you use views where appropriate. Note that some views 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), consider only views that can be updated. In most cases this makes the program easier to read and modify.

Updating index columns

When you update index columns, you must consider two things.

- 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 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.

If the index is changed and the uniqueness is dropped, the program does not execute correctly when two or more rows are returned. The program receives an SQL error code.

Commit processing

CICS ignores any EXEC SQL COMMIT statements 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 end of task.

Be aware of the actions that happen 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 might 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 high.

About this task

You must allow serialization only for low-volume transactions because of potential queuing 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 points.

- 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, 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 in the CICS Db2 environment.

CICS and CURSOR WITH HOLD option

Use CURSOR WITH HOLD in your CICS program to keep the cursor open and in position during a SYNCPOINT.

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 remains 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 Db2 attachment facility forces signon to restore the thread to the initial state, and this incurs additional processor time.

Note:

If the feature toggle `com.ibm.cics.db2.sharelocks=true` is enabled in a CICS region, CICS can pass an XID to Db2 and instruct Db2 to share locks between threads that pass the same XID. The passing of an XID involves a partial signon to Db2 for each unit of work (UOW). This action closes cursors, so held cursors across syncpoints are not supported. Applications must reposition cursors after a syncpoint.

Using the same XID, other threads that originate from other CICS regions or from other transaction managers such as IMS TM can access Db2 in the same global UOW. This avoids having to deal with UOW affinities. The XID token is not used for recovery between CICS and Db2.

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, because 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.

Avoiding AEY9 abends

An AEY9 abend occurs if an application issues an EXEC SQL command when the CICS Db2 attachment facility has not been enabled.

About this task

You can use the following CICS command to detect whether the CICS Db2 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 Db2 attachment facility is not enabled.

When the CICS Db2 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 Db2 attachment facility is ready to accept SQL requests
- NOTCONNECTED, when the CICS Db2 attachment facility is not ready to accept SQL requests

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

Figure 28 on page 89 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 28. 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 z/OS Workload Manager (WLM) is taking place.

Testing for the availability of Db2 can lead to the "storm drain effect". If an application returns normally when a connection to a workload manager is unavailable, a workload manager can be deluded into routing more work to the CICS region, because it believes that good response times are being achieved. For an explanation of the storm drain effect, see Avoiding the storm drain effect. 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.

Chapter 6. Using JDBC and SQLJ to access Db2 data from Java programs

Java programs that run in CICS can use several methods to access data held in a Db2 database.

Java programs can:

- Use a JCICS LINK command to link to a CICS program that uses Structured Query Language (SQL) commands to access the data.
- Directly access the data by using the Java Data Base Connectivity (JDBC) or Structured Query Language for Java (SQLJ) application programming interfaces.

The IBM Data Server Driver for JDBC and SQLJ

When a Java application for CICS makes JDBC and SQLJ requests of either type 2 connectivity or type 4 connectivity to an IBM Db2 and other programs for Db2 for z/OS database, the requests are processed by the IBM Data Server Driver for JDBC and SQLJ.

In a CICS environment using type 2 connectivity, the IBM Data Server Driver for JDBC and SQLJ converts the JDBC or SQLJ requests into their EXEC SQL equivalents. The converted requests flow into the CICS Db2 attachment facility in the same way as EXEC SQL requests from non-Java programs. The customization and tuning options for the CICS Db2 attachment facility apply equally to Java and non-Java programs.

When you are using type 2 connectivity, Db2 client information such as the ClientUser and ClientHostName values are not supported and are ignored.

The Liberty JVM server also supports type 4 connectivity that uses TCP/IP to connect to the Db2 subsystem instead of the CICS Db2 attachment facility. To use the IBM Data Server Driver for JDBC and SQLJ with CICS, you must be connected to a Db2 subsystem that supports the appropriate level of your driver version.

Details of how to code and build Java applications that use the JDBC and SQLJ application programming interfaces that apply to your version of Db2 can be found in Java application development for IBM data servers.

Configuring a JVM server to support Db2

A JVM server is the runtime environment for Java applications. You can configure the JVM server to support JDBC and SQLJ-based applications.

Before you begin

To use the JVM server with Db2, it is best practice to install the latest version of the IBM Data Server Driver for JDBC and SQLJ. You must add the Db2 SDSNLOD2 library to the CICS STEPLIB concatenation. For more information about required APARs, see Detailed system requirements.

About this task

Enable your applications to use the IBM Data Server Driver for JDBC and SQLJ that is supplied with Db2 with the following steps.

Procedure

1. To set up the driver for a Liberty JVM server, see Configuring a Liberty JVM server. For an OSGi JVM server, see Configuring a JVM server for OSGi applications.

2. The Db2 environment variable **DB2SQLJPROPERTIES** is not supported in a JVM server. Therefore, you must set the properties that are related to the Db2 driver directly in your JVM profile. For a list of available properties, see Programming for Db2 for z/OS in Db2 for z/OS product documentation.
3. Optional: If you want to generate Db2 trace for a JVMSERVER, the following JVM profile properties are useful:

```
-Ddb2.jcc.traceDirectory=/u/<userID>/db2trace/
-Ddb2.jcc.traceFile=jccTrace.txt
-Ddb2.jcc.appendFile=true
-Ddb2.jcc.traceLevel=-1
```

Important: If a Db2 property is set in the JVM profile, it becomes the default for all dataSources that do not otherwise set them.

What to do next

If you want to use JDBC or SQLJ from a Java application in an OSGi JVM server with a Java 2 security policy mechanism active, see Enabling a Java security manager.

Setting the Db2 schema in a JVM server

You can set the current Db2 schema for a JVM server that accesses Db2 in a number of ways.

If the default Db2 schema is not the one that you want, you have two options:

- You can set the `db2.jcc.currentSchema` to the name of the schema you want in the JVM profile. For example, the `-Ddb2.jcc.currentSchema=CICSDB2Setting` schema in the JVM profile ensures that all of the Db2 access through the JCC driver respects the chosen schema. For a Liberty JVM server, all dataSource definitions respect this value, unless they are otherwise overridden in the dataSource definition.
- If you are using a Liberty JVM server, you can set the schema for an individual dataSource. For example:

```
<dataSource id="ds1" jndiName="jdbc/ds1">
    <jdbcDriver libraryRef="db2Lib"/>
    <properties.db2.jcc currentSchema="TESTER"/>
</dataSource>
```

Tip: If you use the CICS default dataSource (created by using autoconfigure), then any values that are set in `server.xml` are overwritten each time the server is started. Create your own dataSource to prevent potential updates to the CICS default dataSource overwriting your values.

Programming to the JDBC and SQLJ APIs

Java programs for CICS must adhere to the programming rules of the JDBC application programming interfaces (API), which might be more restrictive than the general CICS programming model.

The IBM Data Server Driver for JDBC and SQLJ provides support for JDBC 4.2 and earlier versions.

When connecting an OSGi Java program to Db2, add `com.ibm.db2.jcc;resolution:=optional` to the Import-Package in your bundle manifest.

Note: SQLJ using type 2 connectivity is supported in a Liberty JVM server. SQLJ using type 4 connectivity is not supported.

You can find more information about the JDBC APIs at the Oracle JDBC website. For information about developing Java applications that use the JDBC and SQLJ APIs, see Java application development for IBM data servers in Db2 for z/OS product documentation.

Deploying a serialized SQLJ profile in a JVM server

SQLJ provides support for embedded static SQL in Java applications. You deploy a serialized SQLJ profile in an OSGi JVM server or a Liberty JVM server as an alternative to using the JDBC application programming interface.

Before you begin

You must prepare your SQLJ program using the following process: Program preparation for SQLJ programs.

Once the DBRMs have been created they must be bound into a Db2 Plan or Package and a Db2 Entry specifying the Db2 plan defined to CICS. Failing to do this will most probably result in SQL -805 errors or SQLJ defaulting to use JDBC. Download the serialized profile file to the workstation where the bundle is being created. Ensure that the transfer is done using the binary format, so that there is no code page conversion. This will prevent problems later.

Now a decision is needed about where to put the serialized profile file. There are two scenarios:

Procedure

1. Scenario one - keep the serialized profile within the deployed bundle.
   a) For OSGI bundles the serialized profile can be kept in the bundle root directory or in the bundles class directory. For Liberty bundles it must go into the bin directory with the other classes.
2. Scenario two - externalize the profile.
   a) Move the serialized profile from within the jar to a directory in the USS file system, for example `/usr/lpp/cicsts/dev/sqlj.profile.dir` . The file needs to be in a directory with the same package structure such as `/usr/lpp/cicsts/dev/sqlj.profile.dir/com/ibm/cics/test/sqlj/CurrentTimeStamp_SJProfile0.ser`.
   b) Add an entry to the bundle manifest.

   ```
   Bundle-ClassPath: .,external:$sqlj.profile.dir$
   ```

   The string 'external' indicates a location external to the bundle and the string enclosed with **$** substitutes the value of a Java system property, that is `sqlj.profile.dir`.
   c) Add a Java system property similar to this example in the JVM profile:

   ```
   -Dsqlj.profile.dir=/usr/lpp/cicsts/dev/sqlj.profile.dir
   ```

Results

You have successfully deployed a serialized SQLJ profile.

Acquiring a connection to a database

Before executing SQL statements, a JDBC or SQLJ application must acquire a connection to a database. The application connects to a target data source using one of two Java interfaces.

About this task

The two Java interfaces are:

- DriverManager: This class connects an application to a database which is specified by a database URL.
- DataSource: This interface is preferred over DriverManager because it allows details about the underlying database to be transparent to your application making applications more portable. The DataSource implementation is created externally to the Java program and obtained with a Java Naming and Directory Interface (JNDI) DataSource name lookup. The DataSource interface is only available in a Liberty JVM server.

In an OSGi JVM server only type 2 connectivity using DriverManager is supported for use with Db2.

In a Liberty JVM server, you can use the DriverManager or DataSource interfaces to access Db2.

If you are using JDBC type 2 connectivity, you do not need to specify a user ID and password, as the existing CICS Db2 security procedures are used instead. It is also advisable to use a default URL, for more detail, see Committing a unit of work.

There are JDBC examples for both OSGi and Liberty JVM servers supplied in the IBM CICS SDK for Java. For more information, see Developing Java applications to run in a Liberty JVM server.

For more information on using the JDBC DriverManager and DataSource interfaces to acquire a connection, and sample code that you can use in your application, see the Db2 for z/OS : Programming for Java which is appropriate for your version of Db2.

Acquiring a DriverManager connection to a database

Using the DriverManager class, the connection to the database is identified by a database Uniform Resource Locator (URL) that is provided to the JDBC driver.

About this task

A `DriverManager` connection is configured for an OSGi JVM server by specifying the Db2 JDBC jars and native DLLs in the JVM profile `OSGI_BUNDLES` and `LIBPATH_SUFFIX` JVM profile.

In a Liberty JVM server, the `DriverManager` configuration is provided by the CICS JDBC Liberty feature.

The JDBC driver recognizes two types of URL:

Default URL
A default URL does not include the location name of a Db2 subsystem. A default URL for Db2 for z/OS can be specified in one of two formats:

```
jdbc:db2os390sqlj:
or
jdbc:default:connection
```

When a default URL is specified, the application is given a connection to the local Db2 to which CICS is connected. If your installation uses Db2 data sharing, you can access all the data in your sysplex from the local Db2.

Explicit URL
An explicit URL includes the location name of a Db2 subsystem. The basic structure of an explicit URL for Db2 for z/OS is:

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

Typically, the location name is the name of the local Db2 to which CICS is connected. However, you can specify the name of a remote Db2 to access. In this case, CICS uses the local Db2 as a pass-through, and uses Db2 Distributed Data Facilities to access the remote Db2.

It is advisable to use a default URL when accessing Db2 with type 2 connectivity. The use of an explicit URL might cause specific actions when a connection closes that could be inconvenient when multiple programs are used in the same application suite. Also, when a default URL is used, the behavior of the connection is not affected by the JDBC driver version.

To acquire a connection, your Java application needs to invoke the `getConnection()` method with the URL. For example:

```
Connection connection =
DriverManager.getConnection("jdbc:default:connection");
```

When connecting an OSGi Java program to Db2, add `com.ibm.db2.jcc;resolution:=optional` to the Import-Package in your bundle manifest.

Acquiring a DataSource connection to a database

Using the DataSource interface, the connection to the database is obtained with a Java Naming and Directory Interface (JNDI) DataSource name lookup. JDBC DataSource implementation is provided by either the `cicsts:jdbc-1.0`, the Liberty `jdbc-4.1` or `jdbc-4.2` feature.

About this task

The term data source can be used in two different contexts, careful attention is needed, as they are distinguished by capitalization:

- A data source is a source of data such as a database.
- A DataSource is the Java EE way of encapsulating a set of properties that identify and describe the real world data source that it represents. A DataSource object can be thought of as a factory for connections to the particular data source that it represents.
- If a DataSource object is registered with a JNDI naming service, an application can use the JNDI API to access that DataSource object, which can then be used to connect to the data source it represents.
- A DataSource object can choose to implement a connection pool (a set of temporary logical representations of a physical connection). When the application closes such a connection, the temporary connection (also known as a JDBC connection) is not closed, but returned to a pool for re-use.

If you are using JDBC type 4 connectivity, and want to coordinate database updates with the updates made in the CICS unit of work, use the CICS JTA integration support and perform your updates within the scope of a UserTransaction. See Java Transaction API (JTA) for more information. Additionally you should ensure that your DataSource is an XADataSource (specified by adding the element `type="javax.sql.XADataSource"` to your definition). If you use the Liberty default DataSource `<dataSource id="DefaultDataSource">` it is an XADataSource by default.

After the DataSource is configured, the application can use the use the value of the `jndiName` specified in either the `cicsts_dataSource` element, or the `dataSource` element, respectively, in the Liberty server configuration to obtain an instance of that `DataSource` class from the JNDI naming service. The `getConnection()` method can then be called on that `DataSource` object to get a connection. For example:

```
Context context = new InitialContext();
DataSource dataSource = (DataSource)
context.lookup("jdbc/defaultCICSDataSource");

Connection connection = dataSource.getConnection();
```

JDBC and SQLJ connection considerations

When using the IBM Data Server Driver for JDBC and SQLJ in CICS Java applications, the following information should be considered.

Using the IBM Data Server Driver for JDBC and SQLJ with type 2 connectivity

If you are using type 2 connectivity, a Java application for CICS can only have one JDBC connection or one SQLJ connection open at once.

Although JDBC allows an application to have multiple connections at the same time, use of type 2 connectivity restricts this to one. However, an application can close an existing connection and open a connection to a new Db2 location, if required.

As a guideline, an application that has an open connection should close the connection before linking to another application that might use JDBC or SQLJ. For Java programs that are part of an application suite, you need to consider the implications of closing the connection, because if you are using an explicit URL, closing the connection can cause a syncpoint to be taken. If you are using a default URL, a syncpoint does not have to be taken when the connection is closed.

See "Committing a unit of work" on page 96 for more information about type 2 connectivity.

Using the IBM Data Server Driver for JDBC and SQLJ with type 4 connectivity (Liberty JVM server only)

If you are using type 4 connectivity, a Java application for CICS can have multiple JDBC connections or multiple SQLJ connections open at once. Care should be taken to maintain data integrity. It is advisable to use the Java Transaction API to clearly demarcate transaction boundaries.

Note:

The JDBC 4.0, 4.1 and 4.2 implementations reside in the same driver.

When you use the `jdbc-4.1` or the `jdbc-4.2` feature, the **`RowSetFactory`** interface and **`RowSetProvider`** class attempt to create a new JDBC connection.

Closing a connection to a database

When you close a connection to a database, JDBC and SQLJ resources are automatically released. Typically, a connection to a database is closed when the task ends.

For performance reasons, an application might leave the JDBC connection open for a subsequent user of the same JVM to use. If the JDBC connection is left open, the application must ensure that JDBC resources are not leaked over time.

If the application leaves the JDBC or SQLJ connection open, the application must do the following:

- Ensure that JDBC and SQLJ resources are released.
- Recover following a Db2 SIGNON for the underlying Db2 connection.
- Recycle the cached connection if it become invalid; for example, StaleConnection, SQLCODE=4499.

If connections are cached, you are recommended to include logic in your application that recycles the connection after a given number of transactions. Recycling cached connections protects against resource leakage.

Committing a unit of work

When using type 2 connectivity, your JDBC and SQLJ applications can issue JDBC and SQLJ commit and rollback method calls. The IBM Data Server Driver for JDBC and SQLJ converts these calls into a JCICS commit or a JCICS rollback call, resulting in a CICS syncpoint being taken.

About this task

A JDBC or SQLJ commit results in the whole CICS unit of work being committed, not just the updates made to Db2. CICS does 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, and this has the same result as issuing a JDBC or SQLJ commit or rollback method call. The whole unit of work is committed or rolled back together, both Db2 updates and updates to CICS controlled resources.

When you are working with JDBC connections, there are some circumstances in which you cannot avoid a syncpoint being taken, and the unit of work being committed, when the connection to the Db2 database is closed. This applies in either of the following circumstances:

- You have used the autocommit property of a JDBC connection.
- You have acquired the DriverManager connection using an explicit URL.

For a stand-alone application, these rules do not cause a problem, as CICS ensures that an end of task syncpoint is taken in addition to any syncpoint that is taken when the connection is closed. However, the JDBC and SQLJ application programming interfaces do not support the concept of multiple application programs for each unit of work. If you have a number of programs that make up an application, one program might access Db2, then call another program that also accesses Db2, in the course of a single unit of work. If you want these programs to be Java programs that use JDBC or SQLJ, you need to ensure that the unit of work is not committed when the connection to Db2 is closed, or else the application will not operate as planned. You should be particularly aware of this requirement if you are replacing programs in an existing application with Java programs that use JDBC or SQLJ, and you want to share the same CICS-Db2 thread between the programs. To address this issue, use a DriverManager with default URL or a DataSource connection.

Autocommit

JDBC applications can use the autocommit property of a JDBC connection. The autocommit property causes a commit after each update to Db2. When using type 2 connectivity, this commit is a CICS ® commit, and results in the whole unit of work being committed.

Using the autocommit property also causes a commit to be taken when a connection is closed. The use of autocommit in with type 2 connectivity is not recommended. For this reason the IBM Data Server Driver for JDBC and SQLJ sets a default of `autocommit(false)` when it runs in a CICS environment with type 2 connectivity. For type 4 connectivity the default is `autocommit(true)`.

Syncpoint issues for DriverManager with explicit and default URLs

When a Java application for CICS that uses JDBC or SQLJ acquires a connection using an explicit URL, it operates in an environment similar to that of a DPL server program linked to with the SYNCONRETURN attribute.

When an application program that uses JDBC or SQLJ closes the explicit URL connection, if CICS is using the IBM Data Server Driver for JDBC and SQLJ, no implicit syncpoint is taken.

However, the close of an explicit URL connection is only successful when on a unit of work boundary. The application must therefore take a syncpoint, by issuing a JDBC or SQLJ commit method call or a JCICS commit, before closing the connection. (The application could use autocommit(true) to ensure that a syncpoint is taken, but the use of this property is discouraged in the CICS environment.) When the application program closes an explicit URL connection, that is the end of the unit of work.

You can overcome this restriction by acquiring the connection using a **default** URL instead of an explicit URL, or by using a data source that provides a default URL connection (see “Acquiring a connection to a database” on page 93). When a default URL is used, the Java application does not have to close the connection on a unit of work boundary, and no syncpoint is taken when the connection is closed (provided that autocommit(true) has not been specified).

It is recommended that you always use default URL connections when using DriverManager type 2 connectivity.

CICS abends during JDBC or SQLJ requests

CICS abends issued during processing of an EXEC SQL request, built by the IBM Data Server Driver for JDBC and SQLJ, 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 7. Preparing CICS Db2 programs for execution and production

This section discusses program preparation in a CICS Db2 environment.

For information on support for Java programs in the CICS Db2 environment, see Using JDBC and SQLJ to access Db2 data from Java programs .

These topics contains diagnosis, modification or tuning information.

The CICS Db2 test environment

When setting up your CICS Db2 test environment, consider how many CICS systems you want to connect to your Db2 systems.

You can connect more than one CICS system to the same Db2 system. However, the CICS Db2 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 connected to two or more different Db2 systems

The first alternative, using a single CICS system for both production and test, is not recommended because applications in test could affect the performance of the production system.

The second alternative, with just one Db2 system, could be used for both test and production. Whether it is suitable depends on the development and production environments involved. Running a test CICS system and a production CICS system separately allows test failures without impacting production.

The third alternative, with, for example, one test and one production Db2 system, is the most flexible. 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.

CICS Db2 program preparation

You can prepare your CICS Db2 program by using the Db2 Interactive (DB2I) interface, or you can submit your own JCL for batch processing.

About this task

The steps shown in Figure 29 on page 100 summarize how to prepare your program for execution after your application program design and coding is complete.

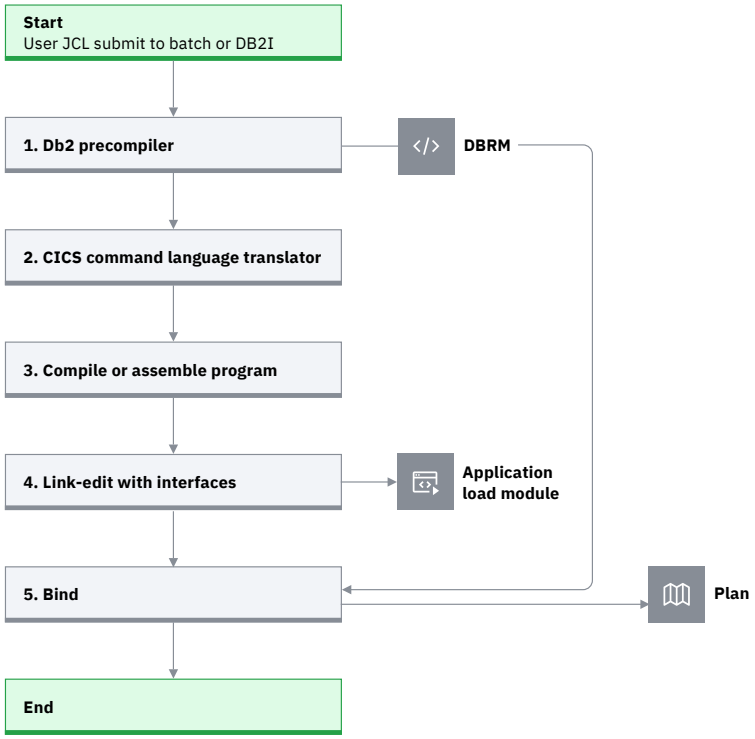


Figure 29. Steps to prepare a CICS application program that accesses Db2

For an overview of the stages in this process, see Preparing a CICS application program that accesses Db2.

When you prepare CICS application programs that access Db2:

- The Db2 precompiler (Step 1) builds a DBRM that contains information about each of the program's SQL statements. It also validates SQL statements in the program. For more information about using the Db2 precompiler, see Programming for Db2 for z/OS in Db2 for z/OS product documentation.
- If the source program is written in PL/I, the input to Step 1, the Db2 precompiler, is the output from the PL/I macro phase (if used).
- You can run Step 1, the Db2 precompiler, and Step 2, the CICS command language translator, in either sequence. The sequence shown is the preferred method, and it is the method supported by the DB2I program preparation panels. If you run the CICS command language translator first, it produces a warning message for each EXEC SQL statement it encounters, but these messages do not affect the result.
- If you use one of the Language Environment-conforming compilers (COBOL and PL/I) that has integrated the CICS translator, translation of the EXEC CICS commands (Step 2) takes place during program compilation (Step 3). For more information on the integrated CICS translator and the compilers that support it, see Translation and compilation.
- If you are running DB2 Version 7 or later and preparing a COBOL or PL/I program using one of the Language Environment-conforming COBOL or PL/I compilers, the compiler also provides an SQL statement coprocessor (which produces a DBRM), so you do not need to use the separate Db2 precompiler (Step 1). See Programming for Db2 for z/OS in Db2 for z/OS product documentation for more information on using the SQL statement coprocessor.
- If you are running DB2 Version 6 or earlier and preparing a COBOL or PL/I program, use the separate Db2 precompiler. For a COBOL program, ensure that you specify a string delimiter that is the same for the Db2 precompiler and the integrated CICS translator. The default delimiters are not compatible.

- In the link edit of the program (Step 4), include both the appropriate CICS EXEC interface module, or stub, for the language in which you are coding, and the CICS Db2 language interface module DSNCLI. The CICS EXEC interface module *must* be included first in the load module. You can link DSNCLI with your program in either 24-bit or 31-bit addressing mode (AMODE=31). If your application program runs in 31-bit addressing mode, you should link-edit the DSNCLI stub to your application with the attributes AMODE=31 and RMODE=ANY so that your application can run above 16MB.
- The bind process (Step 5) requires Db2. The bind process uses the DBRM to produce an application plan (often just called a plan) that enables the program to access Db2 data. See The bind process for more information on the bind process. A group of transactions that use the same entry thread (in other words, specified in the same DB2ENTRY) must use the same application plan. Their DBRMs must be bound into the same application plan, or bound into packages that are then listed in the same application plan.

Table 8 on page 101 shows the tasks that you need to perform to prepare a CICS Db2 program, depending on the language of the program and on your version of Db2:

Table 8. Tasks to prepare a CICS program that accesses Db2

Db2 version and program language	Step 1 (SQL statement processing)	Step 2 (CICS command translation)	Step 3 (Program compile)	Step 4 (Link-edit)	Step 5 (Bind)
DB2 Version 6 and Assembler	Db2 precompiler	CICS-supplied separate translator	Language compiler	Link-edit with EXEC interface and DSNCLI	Bind process
DB2 Version 6 and PL/I	Db2 precompiler	Language compiler that supports integrated CICS translator		Link-edit with EXEC interface and DSNCLI	Bind process
DB2 Version 6 and COBOL	Db2 precompiler	Language compiler that supports integrated CICS translator		Link-edit with EXEC interface and DSNCLI	Bind process
DB2 Version 6 and other languages	Db2 precompiler	CICS-supplied separate translator	Language compiler	Link-edit with EXEC interface and DSNCLI	Bind process
DB2 Version 7 (or later) and Assembler	Db2 precompiler	CICS-supplied separate translator	Language compiler	Link-edit with EXEC interface and DSNCLI	Bind process
DB2 Version 7 (or later) and PL/I	Language compiler that supports integrated CICS translator and SQL statement coprocessor			Link-edit with EXEC interface and DSNCLI	Bind process
DB2 Version 7 (or later) and COBOL	Language compiler that supports integrated CICS translator and SQL statement coprocessor			Link-edit with EXEC interface and DSNCLI	Bind process
DB2 Version 7 (or later) and other languages	Db2 precompiler	CICS-supplied separate translator	Language compiler	Link-edit with EXEC interface and DSNCLI	Bind process

You prepare your CICS Db2 program by using the Db2 Interactive (DB2I) interface or by submitting your own JCL for batch execution.

- Db2 Interactive (DB2I) interface: 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 Programming for Db2 for z/OS in Db2 for z/OS product documentation.
- 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 prepare your program for execution while CICS is running, you might need to issue a CEMT NEWCOPY command to make the new version of the program known to CICS.

CICS SQLCA formatting routine

DSNTIAR, the IBM-supplied SQLCODE message formatting procedure, lets you send "SQL messages online" to your application.

With DB2 Version 3.1, DSNTIAR was split into two front-end modules (DSNTIAC and DSNTIAR) and a runtime module (DSNTIA1). DSNTIAC is used for CICS applications and DSNTIAR for other Db2 interfaces. This change removed the need, previous to DB2 Version 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. If you have applications that have previously been link-edited with DSNTIAR, you should consider link-editing them again using DSNTIAC instead, which will provide performance improvements and isolate them from changes to DSNTIAR.

The CICS front-end part, DSNTIAC, is supplied as a source member in the Db2 library SDSNSAMP.

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.

What to bind after a program change

If you change a program, you must prepare it and rebind it before it can be used.

About this task

For an overview of the bind process, see The bind process. For an overview of plans and packages, see Plans, packages and dynamic plan exits.

Imagine you have a CICS transaction that consists of four program modules: module 1 is the main module. Module 1 calls module 2. Module 1 also calls module 3, and module 3 calls module 4. It is not unusual that the number of modules is high in a real transaction. Assuming that at least one SQL statement changed in one of the modules, you must perform the following procedure to prepare the program and to make the transaction executable again.

Procedure

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. **If the DBRM for program C was bound into a package**, bind that package using the new DBRM, and all the application plans that use program C will automatically locate the new package.
6. **If the DBRM for program C was bound directly into any application plans**, locate all the application plans that include the DBRM for program C. Bind all the application plans again, using the DBRMs for all the programs directly bound into them, to get new application plans. 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.

   Note: If you have not used packages before, note that using packages simplifies the rebinding process. You can 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 then use the BIND PACKAGE command to bind the DBRMs for any changed programs, instead of using the BIND PLAN command to bind the whole application plan. This provides increased transaction availability and better performance. See Using Db2 packages for more information on using packages.

Bind options and considerations for programs

When binding multiple programs into an application plan, be aware of the way in which Db2 uses time stamps.

For each program, the Db2 precompiler creates the following:

- The Db2 precompiler creates a DBRM with a time stamp of Td *x*. For example, Td1 for the first program, Td2 for the second program, and so on.
- The Db2 precompiler creates a modified source program with a time stamp of Ts *x* in the SQL parameter list. For example, Ts1 and Ts2, if two programs are involved.

At bind time, the DBRM for each program is bound into the package or plan that you have specified. In addition, Db2 updates its catalog table SYSIBM.SYSDBRM with one line for each DBRM, together with its time stamp. At execution time, Db2 checks the time stamps for each SQL statement, and returns a -818 SQL code if the time stamp for the DBRM and the time stamp it has placed in the source program are different (in our example, if Td1 and Ts1 are different, or Td2 and Ts2 are different). To avoid -818 SQL codes, use one of the following strategies:

- Bind all programs into packages, and list these packages in the application plan. When a program changes, precompile, compile, and link-edit the program, and bind it into a package again.
- If you bind any programs directly into application plans, ensure that for every new or changed program, you precompile, compile, and link-edit the program, then bind all the application plans that involve that program, using the DBRMs from all the programs directly bound into those plans. Use the BIND command, not the REBIND command, to do this.

When you bind a plan, a number of options are available. Almost all bind options are application dependent and should be taken into account during the application design. 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.

RETAIN

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.

Isolation level

The isolation level is specified for the complete plan. You are recommended to use Cursor Stability (CS) unless there is a specific need for using Repeatable Read (RR). Using CS allows a high level of concurrency and reduces 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 ACQUIRE and RELEASE parameters change from plan to plan over time because they are related to the transaction rate.

The general recommendations for these parameters are described in Selecting BIND options for optimum performance.

CICS Db2 program testing and debugging

The tools that are typically used in a CICS environment can be used to test and debug a CICS application program that accesses Db2. These include: the execution diagnostic facility (EDF), the CICS auxiliary trace, and transaction dumps.

For information about these and other problem determination processes, see Troubleshooting Db2.

Going into production: checklist for CICS Db2 applications

This checklist shows the tasks you need to perform after designing, developing, and testing an application, in order to put the application into production.

About this task

These tasks are highly dependent on the standards you have used in the test system. 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.

Prepare 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 for the production system

If you have bound your programs into packages on the test system, you do not need to perform this step. You can move the packages straight to the production system. See "Produce an application plan for the production system" for details of how to do this. However, if you want to bind your programs directly into application plans, or if you want to bind the programs into packages on the production system, you need to put the DBRMs for the programs on the production system. You can either:

- 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 for the production system

To produce load modules:

- If the DBRMs were produced by precompiling on the production system, compile and link-edit the CICS modules on the production system; or
- If the DBRMs were copied, or if you are moving packages from the test system to the production system, copy the load modules from the test system to the production system libraries.

Table 9 on page 105 shows a procedure you can use to copy a changed load module from a test system to a production system library, replacing the old version of the load module.

Table 9. Moving a changed program from the test environment to the production environment

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 the old version or the new version of the program. Then the correct version of the package is run, determined by the consistency token embedded in the program load module.

Produce an application plan for the production system

If you have bound your programs into packages on the test system, you can copy the packages to the production system, including them in a collection that is listed in the application plan. When you copy a package to the production system, you do not need to bind the application plan on the production system again, as long as the package is included in a collection that is already listed in the application plan.

Table 10 on page 105 shows a procedure you can use to copy a changed package from a test system to a production system library, replacing the old version of the package. This example uses the VERSION keyword at precompile time to distinguish the different versions of the packages. For a full explanation and use of the VERSION keyword, see Programming for Db2 for z/OS in Db2 for z/OS product documentation.

Table 10. Moving a changed package from the test environment to the production environment

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 on the test system and then copied to the production system

Table 10. Moving a changed package from the test environment to the production environment (continued)		
Test system	**Production system**	**Notes**
	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

If you want to bind your programs directly into application plans, or if you want to bind the programs into packages on the production system, you must perform the bind process on the DBRMs that you have placed on the production system. If you are binding your programs directly into application plans, you must then bind all the application plans on the production system that involve those programs. See The bind process for more information on the bind process. Note that 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 a plan on the production system, to check the Db2 optimizer decisions.

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 definitions
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. 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.

In addition, if RACF is installed, you need to define new users and Db2 objects.

Tuning a CICS application that accesses Db2

CICS applications that access Db2 must be tuned before they move to production and periodically whist they are in production.

About this task

When moving a CICS application that accesses Db2 to production, add these checks to those already performed for CICS :

- Check that all the application programs that make Db2 requests are threadsafe. If they are, you will be exploiting the open transaction environment (OTE), and improving the performance of the application. See Enabling CICS Db2 applications to use OTE through threadsafe programming for an explanation of how application programs work in the open transaction environment.
- 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 a CICS application that accesses Db2 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.

Chapter 8. Monitoring Db2

Accounting and monitoring in a CICS Db2 environment.

CICS-supplied accounting and monitoring information

CICS includes several facilities that provide accounting and monitoring. You can use these facilities to measure the use of different resources in a CICS system.

The most frequently used tools are as follows:

- Statistics data.

  CICS statistics are the simplest tool for monitoring a CICS system. They contain information about the CICS system as a whole, such as its performance and use of resources. CICS statistics are suitable for performance tuning and capacity planning. CICS collects statistics during online processing, and processes them offline later. The statistics domain collects this data and then writes records to the System Management Facility (SMF) data set provided by MVS. The records are of SMF type 110. You can process these records offline using the DFHSTUP program.

- Monitoring 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 of 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 the following classes of data:

  - Performance class for detailed transaction level information
  - Transaction resource data for additional transaction-level information about individual resources accessed by a transaction
  - Exception class for exception conditions

CICS is a multitasking address space, and CICS monitoring facilities are generally used to determine the processor time and other resources that are consumed by the individual transactions, or functions performed by CICS.

For a full description of the CICS monitoring facilities, and details on activating, collecting, and processing this information, see CICS monitoring facility: Performance and tuning .

For both statistics data and monitoring data, you can use an offline processing facility. CICS Performance Analyzer and IBM Z® Decision Support are two tools that collect and analyze data from CICS and other IBM systems and products. They can build reports that help you with:

- Systems overview
- Service levels
- Availability
- Performance and tuning
- Capacity planning

For more information, see CICS Performance Analyzer for z/OS (CICS PA) and IBM Z Decision Support in Improving performance.

Db2-supplied accounting and monitoring information

The instrumentation facility component of Db2 enables you to use six types of traces: statistics, accounting, audit, performance, monitor, and global. 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.

Statistics

Describe the total work executed in Db2. This information is not related to any specific 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 can make it difficult to directly relate the activity to specific 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 signon, 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.

Monitoring a CICS Db2 environment: Overview

The objective of monitoring the CICS Db2 attachment facility is to provide a basis for accounting and tuning.

About this task

You can use monitoring 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.

It is recommended that you monitor your test environment for the following reasons:

- To check that new programs function correctly (that is, use the correct call sequence) against test databases.
- To detect any performance problems due to excessive I/O operations or inefficient SQL statements.
- To detect bad design practices, such as holding Db2 resources across screen conversations.
- To set up optimum locking protocols to balance the application isolation requirements with the requirements 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.

You can use some, or all, of the following tools to monitor the CICS Db2 attachment facility and CICS transactions that access Db2 resources.

- Monitor the CICS Db2 attachment facility using:
  - CICS Db2 attachment facility commands
  - Db2 commands
  - CICS Db2 statistics

  See “Monitoring the CICS Db2 attachment facility” on page 112.
- Monitor CICS transactions using:
  - CICS monitoring facility (CMF)
  - CICS auxiliary trace

  See “Monitoring CICS transactions that access Db2 resources” on page 115.
- Monitor Db2 using:
  - Db2 statistics records
  - Db2 accounting records
  - Db2 performance records

  See “Monitoring Db2 when used with CICS” on page 116.
- Monitor the CICS system (for example, the dispatcher) with CICS statistics. See “Monitoring the CICS system in a CICS Db2 environment” on page 118.

Monitoring the CICS Db2 attachment facility

Monitor the CICS Db2 attachment facility by using commands addressed to both Db2 and the CICS Db2 attachment facility itself.

About this task

Monitoring the CICS Db2 attachment facility using CICS Db2 attachment facility commands

You can monitor the status of CICS-Db2 threads, and the corresponding CICS transactions using them, with the DSNC DISPLAY PLAN or TRAN command provided by the CICS Db2 attachment facility.

You can also use commands provided by CICS, such as the CEMT or **EXEC CICS INQUIRE** command, on the DB2CONN or on individual DB2ENTRYs. Used on the DB2CONN, the commands enable you to monitor the status of the overall connection between CICS and Db2, as well as the use of the pool. Used on an individual DB2ENTRY, the commands enable you to monitor the use of the DB2ENTRY.

For more information, see CICS-supplied transactions for CICS Db2.

Monitoring the CICS Db2 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.

About this task

The commands take the following form:

```
DSNC-DB2command
```

For example, the DSNC -DIS THREAD command can show 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 Db2 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-supplied transactions for CICS Db2.

Monitoring the CICS Db2 attachment facility using 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.

About this task

The CICS Db2 global and resource statistics are described in detail in CICS Db2 statistics in Reference.

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.

Procedure

1. Use one of the following methods to collect Db2 statistics.
   - Use the **EXEC CICS COLLECT** statistics command with the DB2CONN keyword to allow CICS Db2 global statistics to be collected. CICS Db2 global statistics are mapped by the DFHD2GDS DSECT.
   - Use the **EXEC CICS COLLECT** statistics command with 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.
   - Use the **EXEC CICS PERFORM STATISTICS** command with the Db2 keyword to allow the user to request that CICS Db2 global and resource statistics to be written out to SMF.

   Alternatively you can use the CICS sample statistics program DFH0STAT to collect 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 Figure 30 on page 114.
2. When you have collected the statistics, the important fields to look at as regards performance and tuning are:
   a) 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 reported for each DB2ENTRY, and (in the Db2 Connection statistics) for the pool. Thread reuse is good for performance, because it avoids the overhead of creating a thread for each CICS transaction or each unit of work. The use of protected threads can increase thread reuse.
   b) 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.
   c) In the Db2 Connection statistics, check the field “Peak number of tasks on Pool Readyq”, and also the field “Peak number of Tasks on TCB Readyq”. If the latter is nonzero, tasks were queued waiting for a Db2 connection to use with their open TCBs, rather than waiting for a thread. The tasks were queued because the TCBLIMIT, the maximum number of TCBs that can be used to control threads into Db2, had been reached. This shows that the number of threads available (the sum of the THREADLIMIT values for the pool, for command threads and for all DB2ENTRYs) exceeds the number of TCBs allowed. TCBLIMIT or the THREADLIMIT values should be adjusted in this case.

Example

```
Applid IYK2Z2G1  Sysid JOHN  Jobname CI13JTD5    Date 09/09/2001  Time 10:38:50                  CICS
6.2.0                PAGE     2

DB2 Connection

  DB2 Connection Name. . . . . . . . . . . :  RCTJT
  DB2 Group ID . . . . . . . . . . . . . . :                          Resync Group Member. . . . . . . . . . . : N/A
  DB2 Sysid. . . . . . . . . . . . . . . . :  DE2D
  DB2 Release. . . . . . . . . . . . . . . :   6.2.0

  DB2 Connection Status. . . . . . . . . . :  CONNECTED            DB2 Connect Date and Time . . . :  09/09/2001
10:37:19.21354
  DB2 Connection Error . . . . . . . . . . :  SQLCODE
  DB2 Standby Mode . . . . . . . . . . . . :  RECONNECT

  DB2 Pool Thread Plan Name. . . . . . . . :
  DB2 Pool Thread Dynamic Plan Exit Name . :  DSNCUEXT

  Pool Thread Authtype . . . . . . . . . . :  USERID               Command Thread Authtype. . . . . . . . . :  N/A
  Pool Thread Authid . . . . . . . . . . . :                       Command Thread Authid. . . . . . . . . . :  JTILLI1

  Signid for Pool/Entry/Command Threads. . :  SSSSSSSS

  Create Thread Error. . . . . . . . . . . :  ABEND                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 . . . . . . . :  NONE

  Current TCB Limit. . . . . . . . . . . . :        100
  Current number of TCBs . . . . . . . . . :         10
  Peak number of TCBs. . . . . . . . . . . :         10

  Current number of free TCBs. . . . . . . :          5

  Current number of tasks on TCB Readyq. . :          0
  Peak number of tasks on TCB Readyq . . . :          0

  Pool Thread Limit. . . . . . . . . . . . :          3           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 Partial
Signons . . . :         0
  Number of Pool Thread Waits. . . . . . . :          0           Number of Pool Thread
Commits . . . . . . . :         0
                                                                  Number of Pool Thread
Aborts. . . . . . . . :         0
  Current number of Pool Tasks . . . . . . :          0           Number of Pool Thread Single
Phase. . . . . :        0
  Peak number of Pool Tasks. . . . . . . . :          0           Number of Pool Thread
Reuses. . . . . . . . :         0
  Current Total number of Pool Tasks . . . :          0           Number of Pool Thread
Terminates. . . . . . :         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
Overflows . . :         0
```

Figure 30. Example output from DFH0STAT: the Db2 Connection report

```
Applid IYK2Z2G1  Sysid JOHN  Jobname CI13JTD5    Date 09/09/2001  Time 10:38:50                  CICS
6.2.0                PAGE     3

DB2 Entries

  DB2Entry Name. . . . . . . . . . . . . . . :  XP05               DB2Entry Status . . . . . . . . . . . . . :  ENABLED
  DB2Entry Static Plan Name. . . . . . . . . :  TESTP05           DB2Entry Disabled Action. . . . . . . . . :  POOL
  DB2Entry Dynamic Plan Exit Name. . . . . . :                    DB2Entry Deadlock Resolution. . . . . . . :  ROLLBACK

  DB2Entry Authtype. . . . . . . . . . . . . :  N/A              DB2Entry Accounting records by. . . . . . :  NONE
  DB2Entry Authid. . . . . . . . . . . . . . :  JTILLI1
                                                                 Number of Calls using DB2Entry. . . . . . . :
16,500
  DB2Entry Thread Wait Setting . . . . . . . :  WAIT             Number of DB2Entry
Signons. . . . . . . . . . :          10
                                                                 Number of DB2Entry Partial
Signons. . . . . :           0
  DB2Entry Thread Priority . . . . . . . . . :  HIGH             Number of DB2Entry
Commits. . . . . . . . . . :           0
  DB2Entry Thread Limit. . . . . . . . . . . :          10       Number of DB2Entry
Aborts . . . . . . . . . . :           0
  Current number of DB2Entry Threads . . . . :           0       Number of DB2Entry Single Phase . . . . . . :
5,500
  Peak number of DB2Entry Threads. . . . . . :          10       Number of DB2Entry Thread Reuses. . . . . . :
5,031
                                                                 Number of DB2Entry Thread
Terminates. . . . . :         464
  DB2Entry Protected Thread Limit. . . . . . . . . . :         5  Number of DB2Entry Thread Waits/
Overflows . . :          306
  Current number of DB2Entry Protected Threads . . . :         5
  Peak number of DB2Entry Protected Threads. . . . . :         5

  Current number of DB2Entry Tasks . . . . . . . . . :         0
  Peak number of DB2Entry Tasks. . . . . . . . . . . :        28
  Current Total number of DB2Entry Tasks . . . . . . :     5,500

  Current number of Tasks on DB2Entry Readyq . . . . :         0
  Peak number of Tasks on DB2Entry Readyq. . . . . . :        18
```

Figure 31. Example output from DFH0STAT: the Db2 Entries report

Monitoring CICS transactions that access Db2 resources

CICS provides accounting and monitoring facilities for the resources required by CICS transactions in the CICS address space.

About this task

These CICS accounting and monitoring facilities can produce three types of records:

- Performance records that record the resources used by each transaction in the CICS address space.
- Transaction resource data to record additional information about individual resources accessed by a transaction
- Exception records that record shortages of resources.

The CICS performance class monitoring records include the following Db2-related data fields, in the group DFHDATA:

DB2REQCT (180)

The total number of Db2 EXEC SQL and instrumentation facility interface (IFI) requests issued by a transaction.

DB2RDYQW (187)

The elapsed time the transaction waited for a Db2 thread to become available.

DB2CONWT (188)

The elapsed time the transaction waited for a Db2 connection to become available to use with its open TCB.

CICS monitoring is used in the CICS Db2 environment with the Db2 accounting facility, to monitor performance and to collect accounting information.

For more information about matching up CICS performance class records and Db2 accounting records, see “Relating Db2 accounting records to CICS performance class records” on page 122. For more information about calculating processor consumption, see “Accounting for processor usage in a CICS Db2 environment” on page 127.

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 Troubleshooting Db2.

Monitoring Db2 when used with CICS

Use the SMF or GTF records produced by Db2 to monitor Db2 usage with CICS .

About this task

The Db2 performance monitor (DB2PM) program product is useful to provide reports based on:

- Statistics records
- Accounting records
- Performance records

The reports in this topic 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.

Monitoring Db2 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.

About this task

For more information about activating these facilities and directing the output to SMF and GTF, see Issuing commands to Db2 using the DSNC transaction, and Starting GTF for Db2 accounting, statistics and tuning.

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 Managing Db2 performance in Db2 for z/OS product documentation 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 is connected to Db2 (that is, both CICS and TSO). However, the counts obtained while running the CICS Db2 attachment facility in a controlled environment (that is, with CICS as the only subsystem connected, or with limited TSO activity) can be very useful.

Managing Db2 performance in Db2 for z/OS product documentation 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 report. You can use this report to monitor the average CICS transaction. Figure 32 on page 117 shows a small part of the report provided by DB2PM. Refer to Managing Db2 performance in Db2 for z/OS product documentation for additional information on these reports.

```
   LOCATION: DSN710P2              DB2 PERFORMANCE MONITOR (V7)
      GROUP: DSN710P2              STATISTICS REPORT - LONG
     MEMBER: DF2D
  SUBSYSTEM: DF2D
DB2 VERSION: V7                                 SCOPE: MEMBER

SQL DML                       QUANTITY  /SECOND  /THREAD  /COMMIT
---------------------------  --------  -------  -------  -------
SELECT                           1.00     0.00     0.05     0.02
INSERT                           9.00     0.00     0.41     0.21
UPDATE                           0.00     0.00     0.00     0.00
DELETE                           0.00     0.00     0.00     0.00

PREPARE                         17.00     0.00     0.77     0.40
DESCRIBE                        34.00     0.00     1.55     0.79
DESCRIBE TABLE                   0.00     0.00     0.00     0.00
OPEN                            31.00     0.00     1.41     0.72
CLOSE                           26.00     0.00     1.18     0.60
FETCH                          827.00     0.00    37.59    19.23

TOTAL                          945.00     0.00    42.95    21.98


SQL DCL                       QUANTITY  /SECOND  /THREAD  /COMMIT
---------------------------  --------  -------  -------  -------
LOCK TABLE                       0.00     0.00     0.00     0.00
GRANT                            0.00     0.00     0.00     0.00
REVOKE                           0.00     0.00     0.00     0.00
SET HOST VARIABLE                0.00     0.00     0.00     0.00
SET CURRENT SQLID                0.00     0.00     0.00     0.00
SET CURRENT DEGREE               0.00     0.00     0.00     0.00
SET CURRENT RULES                0.00     0.00     0.00     0.00
SET CURRENT PATH                 0.00     0.00     0.00     0.00
SET CURRENT PRECISION            0.00     0.00     0.00     0.00

CONNECT TYPE 1                   0.00     0.00     0.00     0.00
CONNECT TYPE 2                  29.00     0.00     1.32     0.67
RELEASE                          0.00     0.00     0.00     0.00
SET CONNECTION                   0.00     0.00     0.00     0.00

ASSOCIATE LOCATORS               0.00     0.00     0.00     0.00
ALLOCATE CURSOR                  0.00     0.00     0.00     0.00

HOLD LOCATOR                     0.00     0.00     0.00     0.00
FREE LOCATOR                     0.00     0.00     0.00     0.00

TOTAL                           29.00     0.00     1.32     0.67
```

```
SUBSYSTEM SERVICES            QUANTITY  /SECOND  /THREAD  /COMMIT
---------------------------  --------  -------  -------  -------
IDENTIFY                        23.00     0.00     1.05     0.53
CREATE THREAD                   22.00     0.00     1.00     0.51
SIGNON                          39.00     0.00     1.77     0.91
TERMINATE                       57.00     0.00     2.59     1.33
ROLLBACK                         8.00     0.00     0.36     0.19

COMMIT PHASE 1                   0.00     0.00     0.00     0.00
COMMIT PHASE 2                   0.00     0.00     0.00     0.00
READ ONLY COMMIT                 0.00     0.00     0.00     0.00

UNITS OF RECOVERY INDOUBT        0.00     0.00     0.00     0.00
UNITS OF REC.INDBT RESOLVED      0.00     0.00     0.00     0.00

SYNCHS(SINGLE PHASE COMMIT)     35.00     0.00     1.59     0.81
QUEUED AT CREATE THREAD          0.00     0.00     0.00     0.00
SUBSYSTEM ALLIED MEMORY EOT      0.00     0.00     0.00     0.00
SUBSYSTEM ALLIED MEMORY EOM      0.00     0.00     0.00     0.00
SYSTEM EVENT CHECKPOINT          3.00     0.00     0.14     0.07


CPU TIMES                              TCB TIME         SRB TIME       TOTAL TIME          /THREAD          /COMMIT
-----------------------------  ---------------  ---------------  ---------------  ---------------  ---------------
SYSTEM SERVICES ADDRESS SPACE      17:40.602755      1:09.182200     18:49.784954        51.353862        26.274069
DATABASE SERVICES ADDRESS SPACE        6.100449        11.626277        17.726726         0.805760         0.412249
IRLM                                   0.051894      2:43.867972      2:43.919867         7.450903         3.812090
DDF ADDRESS SPACE                      1.195607         0.212343         1.407950         0.063998         0.032743

TOTAL                              17:47.950705      4:04.888792     21:52.839497        59.674523        30.531151
```

Figure 32. Sample statistics report from DB2PM

Figure 32 on page 117 includes information about:

- *SQL DML*. This information can be used to monitor the SQL requests issued.
- *SQL DCL*. This information can be used to check whether the application is using LOCK statements.

- *Subsystem Services*. This section provides information on thread usage and signon activity. For performance reasons, thread reuse is recommended in CICS environments, to avoid the overhead of creating a thread for each CICS transaction.

Further useful information in the statistics reports, not shown in Figure 32 on page 117, is:

- *Locking* can be used to monitor the number of timeouts and deadlocks. For more information about deadlocks, see Handling deadlocks in the CICS Db2 environment.
- *Buffer Pool* provides information on:
  - The number of data sets opened (Data sets Opened)
  - The number of pages retrieved (GETPAGE Requests)
  - Number of I/Os (Read Operations and Write I/O Operations)

This statistical data is not checkpointed and is not retained by Db2 across restarts.

Monitoring Db2 using the Db2 accounting facility

The Db2 accounting facility output for a single transaction can be used for monitoring and tuning purposes.

About this task

For information on using the Db2 accounting facility, see "Db2 accounting reports" on page 121.

Monitoring Db2 using the Db2 performance facility

The Db2 performance facility trace provides detailed information on the flow of control inside Db2.

About this task

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 the CICS system in a CICS Db2 environment

You can monitor the activity of the CICS Db2 TCBs using the CICS dispatcher statistics.

About this task

For more information about using the statistics provided by the CICS dispatcher, see CICS dispatcher: performance and tuning.

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 Dispatcher domain statistics.

You can obtain the total processor time used by the CICS address space from RMF Monitor II 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:

- Processor time consumed in the DBCTL threads
- Processor time consumed by the IBM MQ threads

These are global performance reports and can help you determine how much of your processor time is being used by CICS.

Accounting in a CICS Db2 environment: Overview

Accounting in a CICS Db2 environment can be used to analyze the transactions being executed in the system, and to charge back the total amount of resources consumed for a given set of transactions to a well-defined set of users. The user can be a real user, groups of users, transactions, or any other expression for the unit to which the resources must be appointed.

Typically the units of consumption include the processor, I/O, and main storage in some weighted proportion. A typical CICS transaction that accesses Db2, consumes resources in the operating system, the CICS system, the application code, and the Db2 address spaces. Each of these components can produce data, which can be used as input to the accounting process. You can combine the output from the different sources to create a complete picture of resource usage for a transaction.

A typical 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 is be the same whenever the transaction is executed. In most cases, this means that the input data to the accounting process must also be repeatable.

When planning the accounting strategy for your CICS Db2 environment, you must:

- Decide the types of Db2 accounting data to use in your accounting process (processor usage, I/O, calls, and so on). “Accounting information provided by the Db2 accounting facility” on page 119 tells you about the accounting data that you can obtain from the Db2 accounting facility.
- Decide how you are going to relate the data from the Db2 accounting record for each transaction to the CICS performance class data for that transaction, to create a complete picture of resource usage for the transaction. “Relating Db2 accounting records to CICS performance class records” on page 122 tells you how you can match up the two types of data.
- Decide whether you are going to relate the CICS performance records and the Db2 accounting records for each transaction back to the specific user, or whether you are going to 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 user. “Strategies to match Db2 accounting records and CICS performance class records and charge resources back to the user” on page 125 gives suggestions for when each method is most appropriate.

If you have decided to use processor usage as the basis for your accounting, “Accounting for processor usage in a CICS Db2 environment” on page 127 has more information on the different classes of processor time that are reported in the Db2 accounting records, and on how to calculate the total processor time used by a transaction.

Accounting information provided by the Db2 accounting facility

The Db2 accounting facility provides detailed statistics on the use of Db2 resources by CICS transactions. You can use Db2 accounting records as the basis for the 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. For information about activating the Db2 accounting facility and directing the output to SMF and GTF, see Starting SMF for Db2 accounting, statistics, and tuning, and Starting GTF for Db2 accounting, statistics and tuning. See Managing Db2 performance in Db2 for z/OS product documentation for information on the general structure of Db2 SMF and GTF records.

The identification section of each Db2 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, the plan name, and the package name.

The Db2 Performance Monitor (DB2PM) program product provides accounting reports taken from the Db2 accounting records.

Data types in Db2 accounting records

This section provides an overview of the different types of data in Db2 accounting records.

You have several possibilities for defining a 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.

Managing Db2 performance in Db2 for z/OS product documentation has details on the individual fields in the Db2 accounting record.

Processor usage

The processor usage information given in the Db2 accounting record typically shows the majority 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.

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 in a CICS Db2 environment" on page 127 for more information on reporting processor usage in a CICS Db2 environment.

I/O

In a Db2 system, the I/O can be categorized into five 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 shows the number of times that Db2 requested a page from the buffer manager.

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 held in the QBACSWS field of the Db2 accounting record.

The number of set write intents is not related to the actual number of write I/Os from the buffer pools, instead it 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 of SQL call activity are repeatable 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.

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.

Db2 accounting reports

The Db2 Performance Monitor (DB2PM) program product provides accounting reports taken from the Db2 accounting records.

Figure 33 on page 122 and Figure 34 on page 122 show examples of long and short accounting reports for a CICS transaction accessing DB2Db2 resources.

```
   LOCATION: DSN710P2                        DB2 PERFORMANCE MONITOR (V7)                          PAGE: 1-1
      GROUP: DSN710P2                          ACCOUNTING REPORT - LONG                  REQUESTED FROM: NOT SPECIFIED
     MEMBER: DF2D                                                                                   TO: NOT SPECIFIED
  SUBSYSTEM: DF2D                             ORDER: PRIMAUTH-PLANNAME                   INTERVAL FROM: 11/05/01 10:42:31.25
DB2 VERSION: V7                                   SCOPE: MEMBER                                     TO: 11/05/01 10:51:03.70

PRIMAUTH: JTILLI1  PLANNAME: DSNJDBC

ELAPSED TIME DISTRIBUTION                                          CLASS 2 TIME DISTRIBUTION
-------------------------------------------------------------      ------------------------------------------------------------------
APPL   |================================================> 98%      CPU    |=> 3%
DB2    |                                                           NOTACC |=> 2%
SUSP   |=> 2%                                                      SUSP   |==============================================> 95%

AVERAGE       APPL(CL.1)  DB2 (CL.2)  IFI (CL.5)    CLASS 3 SUSPENSIONS   AVERAGE TIME  AV.EVENT    HIGHLIGHTS
------------  ----------  ----------  ----------    -------------------   ------------  --------    --------------------------
ELAPSED TIME   25.435644    0.504442         N/P    LOCK/LATCH(DB2+IRLM)      0.000000      0.00    #OCCURRENCES    :        2
 NONNESTED     25.435644    0.504442         N/A    SYNCHRON. I/O             0.085908      6.50    #ALLIEDS        :        2
 STORED PROC    0.000000    0.000000         N/A     DATABASE I/O             0.085908      6.50    #ALLIEDS DISTRIB:        0
 UDF            0.000000    0.000000         N/A     LOG WRITE I/O            0.000000      0.00    #DBATS          :        0
 TRIGGER        0.000000    0.000000         N/A    OTHER READ I/O            0.042337      1.00    #DBATS DISTRIB. :        0
                                                    OTHER WRTE I/O            0.000000      0.00    #NO PROGRAM DATA:        2
CPU TIME        0.016663    0.015404         N/P    SER.TASK SWTCH            0.352902      4.00    #NORMAL TERMINAT:        2
 AGENT          0.016663    0.015404         N/A     UPDATE COMMIT            0.000000      0.00    #ABNORMAL TERMIN:        0
  NONNESTED     0.016663    0.015404         N/P     OPEN/CLOSE               0.206822      1.50    #CP/X PARALLEL. :        0
  STORED PRC    0.000000    0.000000         N/A     SYSLGRNG REC             0.024259      1.00    #IO PARALLELISM :        0
  UDF           0.000000    0.000000         N/A     EXT/DEL/DEF              0.121821      1.50    #INCREMENT. BIND:        0
  TRIGGER       0.000000    0.000000         N/A     OTHER SERVICE            0.000000      0.00    #COMMITS        :        3
 PAR.TASKS      0.000000    0.000000         N/A    ARC.LOG(QUIES)            0.000000      0.00    #ROLLBACKS      :        0
                                                    ARC.LOG READ              0.000000      0.00    #SVPT REQUESTS  :        0
SUSPEND TIME         N/A    0.481147         N/A    STOR.PRC SCHED            0.000000      0.00    #SVPT RELEASE   :        0
 AGENT               N/A    0.481147         N/A    UDF SCHEDULE              0.000000      0.00    #SVPT ROLLBACK  :        0
 PAR.TASKS           N/A    0.000000         N/A    DRAIN LOCK                0.000000      0.00    MAX SQL CASC LVL:        0
                                                    CLAIM RELEASE             0.000000      0.00    UPDATE/COMMIT   :     0.00
NOT ACCOUNT.         N/A    0.007891         N/A    PAGE LATCH                0.000000      0.00    SYNCH I/O AVG.  : 0.013217
DB2 ENT/EXIT         N/A       35.00         N/A    NOTIFY MSGS               0.000000      0.00
EN/EX-STPROC         N/A        0.00         N/A    GLOBAL CONT.              0.000000      0.00
EN/EX-UDF            N/A        0.00         N/A    FORCE-AT-COMMIT           0.000000      0.00
DCAPT.DESCR.         N/A         N/A         N/P    ASYNCH IXL REQUESTS       0.000000      0.00
LOG EXTRACT.         N/A         N/A         N/P    TOTAL CLASS 3             0.481147     11.50
```

Figure 33. Accounting long report for a CICS transaction accessing Db2 resources

```
   LOCATION: DSN710P2                        DB2 PERFORMANCE MONITOR (V7)                          PAGE: 1-1
      GROUP: DSN710P2                          ACCOUNTING REPORT - SHORT                 REQUESTED FROM: NOT SPECIFIED
     MEMBER: DF2D                                                                                   TO: NOT SPECIFIED
  SUBSYSTEM: DF2D                                 ORDER: PLANNAME                        INTERVAL FROM: 11/05/01 10:42:31.25
DB2 VERSION: V7                                   SCOPE: MEMBER                                     TO: 11/05/01 10:50:14.53

                          #OCCURS #ROLLBK SELECTS INSERTS UPDATES DELETES CLASS1 EL.TIME CLASS2 EL.TIME GETPAGES SYN.READ LOCK SUS
PLANNAME                  #DISTRS #COMMIT FETCHES   OPENS  CLOSES PREPARE CLASS1 CPUTIME CLASS2 CPUTIME BUF.UPDT TOT.PREF #LOCKOUT
------------------------- ------- ------- ------- ------- ------- ------- -------------- -------------- -------- -------- --------

DSNJDBC                         1       0    0.00    0.00    0.00    0.00       1.706541       1.003194   249.00    13.00     0.00
                                0       2    4.00    2.00    2.00    2.00       0.027471       0.025984     0.00     5.00        0

TESTP05                         2       0    0.00    0.50    0.00    0.00      33.283119       0.215656     7.00     0.00     0.00
                                0       2    1.50    0.50    0.00    0.00       0.001908       0.001389     1.00     0.00        0

***  GRAND TOTAL  ***
                                3       0    0.00    0.33    0.00    0.00      22.757593       0.478169    87.67     4.33     0.00
                                0       4    2.33    1.00    0.67    0.67       0.010429       0.009587     0.67     1.67        0
```

Figure 34. Accounting short report for a CICS transaction accessing Db2 resources

Relating Db2 accounting records to CICS performance class records

To see a complete picture of the usage of resources by each CICS transaction in both the CICS and Db2 address spaces, you must match up the data in the Db2 accounting records with the data in the CICS performance class records.

About this task

When you have a complete picture of resource usage, you can analyze the resources used by individual transactions, and also charge back to the user the total amount of resources consumed for a given set of transactions, using whatever data types you have decided to include in your cost formula.

If you are examining resource usage because you are carrying out performance analysis, rather than accounting, then you always need to match up the Db2 accounting records and the CICS performance class records. However, if you are examining resource usage for accounting purposes, you might not need to match up the Db2 accounting records and the CICS performance class records.

You do not need to match up the records if you have chosen to use processor time consumption as the basis for your accounting, and you are not using Db2 sysplex query parallelism (parallel query). In this situation, when parallel query is not used, the processor time consumed in Db2 is reported by the CICS performance class records as well as by the Db2 accounting records, so the CICS performance class record for a transaction gives you all the information on processor time that you need to charge the resources for that transaction back to the user. If you are in this situation, skip the rest of this section, and instead read the sections "Accounting for processor usage in a CICS Db2 environment" on page 127 (to understand how processor time consumption is reported), and "Calculating CICS and Db2 processor times for Db2" on page 134 (to find out how to use the information on processor time that is available to you).

If you have chosen to use data as well as, or other than, processor time consumption as the basis for your accounting, read the related concepts to this section to find out how to match up the Db2 accounting records and the CICS performance class records. If you are using processor time consumption for your accounting, you can then read the section "Accounting for processor usage in a CICS Db2 environment" on page 127 to find out how to calculate your processor time consumption using your matched-up Db2 accounting records and CICS performance class records.

What are the issues when matching Db2 accounting records and CICS performance records?

Because CICS and Db2 have different accounting needs, it is not always easy to match up Db2 accounting records and CICS performance class records.

There are two main issues involved when trying to match these records:

1. There is not necessarily a one-to-one relationship between the CICS performance class 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. The Db2 accounting records do not have a field that matches exactly with the corresponding CICS performance records.

For the purpose of charging resources back to users, it is possible to give each user a different authorization ID from a Db2 viewpoint, by specifying the DB2ENTRY or DB2CONN parameter AUTHTYPE as OPID, USERID, GROUP, or TERM. In this case, a Db2 accounting record is generated that contains data only for the authorization ID. You can then collect together all the Db2 accounting records by authorization ID, and charge the resources consumed directly to the user. This method means that you do not need to match the Db2 accounting records with the CICS performance class records. However, from a usability and performance viewpoint, using OPID, USERID, GROUP, and TERM is not an attractive solution, for the reasons discussed in Controlling users' access to plans. For large networks, specifying these authorization IDs can complicate maintenance and result in performance overhead. It is preferable to plan your use of authorization IDs with performance in mind, and assign Db2 accounting records to the end user by matching them to the CICS performance class records.

This section tells you:

- What you can do to make the relationship between Db2 accounting records and CICS performance class records more straightforward.
- What information in a Db2 accounting record can be used to identify the corresponding CICS performance class records.
- What strategies you can use to match Db2 accounting records and CICS performance class records in four typical scenarios.

Controlling the relationship between Db2 accounting records and CICS performance class records

By default, Db2 writes its accounting records at thread termination, or at the signon of a new authorization ID that is reusing the thread.

About this task

If a thread is reused by a transaction that has the same CICS transaction ID and the same Db2 authorization ID as the previous transaction that used the thread, Db2 does not write an accounting record at that point. (See Providing authorization IDs to Db2 for the CICS region and for CICS transactions for information about the relationship between the CICS transaction ID and the Db2 authorization ID.) This means that each Db2 accounting record for the thread can contain information about multiple CICS transactions. In addition, if different types of CICS transactions use the same transaction ID to access Db2, the Db2 accounting record can contain information about different types of CICS transactions.

There are three ways in which you can influence the relationship between Db2 accounting records and CICS performance class records, to deal with these issues:

- You could design your CICS applications so that each CICS transaction ID and Db2 authorization ID always represents the same piece of work, that consumes the same resources. This ensures that each Db2 accounting record contains either a single piece of work, or more than one occurrence of the same piece of work, and it will not contain different items. If such a Db2 accounting record contains multiple items, because the items are identical you can divide the resources used equally between them. However, it might not be practical to design your applications in this way. For example, take the case where a terminal user has a menu displayed at the terminal, and can choose different options (involving different pieces of work) for the next transaction. If the previous transaction ended by setting up the CICS transaction ID with the EXEC CICS RETURN TRANSID(zzzz) command, the next transaction runs under the transaction ID zzzz, no matter what piece of work the terminal user chooses. You might not want to re-design this application for accounting purposes.
- You could avoid reusing threads. This ensures that the thread terminates, and Db2 writes an accounting record, after each task, so each Db2 accounting record represents a single task. However, by doing this, you would lose the significant performance advantages of thread reuse. Also, if different types of transaction used the same transaction ID to access Db2, each Db2 accounting record could still refer to one of several possible tasks.
- You can make Db2 produce an accounting record each time a CICS task finishes using Db2 resources, by specifying ACCOUNTREC(TASK) in the DB2CONN or DB2ENTRY definition. ACCOUNTREC(TASK) is recommended rather than ACCOUNTREC(UOW). This ensures that there is at least one identifiable Db2 accounting record for each task, and the Db2 accounting record will not contain multiple tasks. Also, to solve the issue of different types of transaction using the same transaction ID to access Db2, when you specify ACCOUNTREC(TASK), CICS passes its LU6.2 token to Db2 to be included in the accounting record. You can use this token to match each Db2 accounting record to the relevant CICS transaction. Using ACCOUNTREC(TASK) is generally the most practical and complete solution to control the relationship between Db2 accounting records and CICS performance class records. It carries an overhead for each transaction, but its usefulness for accounting purposes normally outweighs this overhead.

Even if you specify ACCOUNTREC(TASK), note that Db2 can only recognize a single CICS task as long as the task continues to use the same thread. If a transaction contains more than one UOW, assuming that it releases the thread at the end of the UOW, it could use a different thread for each of its UOWs. This can happen with terminal-oriented transactions issuing multiple syncpoints (commit or rollback), and also non-terminal-oriented transactions if NONTERMREL(YES) is set in the DB2CONN. In these cases, Db2 produces an accounting record for each UOW, because it does not recognize them as a single task. So for this kind of transaction, each Db2 accounting record can contain information about only a part of the transaction, and you need to ensure that all the relevant Db2 accounting records for the transaction are identified.

Using data in the Db2 accounting record to identify the corresponding CICS performance class records

Use fields in Db2 accounting records to identify the corresponding CICS performance records.

About this task

The fields in the Db2 accounting records that you might want to use are as follows:

- The CICS LU6.2 token. If you specify either ACCOUNTREC(TASK) or ACCOUNTREC(UOW) 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. The presence of this token makes matching the two sets of records much more simple.
- The thread correlation ID, which contains the CICS 4-character transaction ID. Remember that if you have not specified ACCOUNTREC(TASK) or ACCOUNTREC(UOW), the Db2 accounting record might contain information about more than one transaction that used this ID. If you have specified ACCOUNTREC(TASK) or ACCOUNTREC(UOW), the Db2 accounting record might only contain part of a transaction (a single UOW), and you need to locate the other records relating to the transaction. If different types of CICS transaction use the same transaction ID, you cannot make a positive identification from this item alone.
- The authorization ID field. As for the thread correlation ID, the Db2 accounting record might contain information about more than one transaction that used this authorization ID, or it might only contain part of a transaction. The authorization ID that a CICS transaction uses is determined by the AUTHID or AUTHTYPE parameter in the DB2CONN or DB2ENTRY. If different types of CICS transaction use the same authorization ID, you cannot make a positive identification from this item alone
- The timestamp fields. The start and end time for the thread can help identify the CICS transactions that the Db2 accounting record covers.

Strategies to match Db2 accounting records and CICS performance class records and charge resources back to the user

There is not one ideal way of matching Db2 accounting records and CICS performance class records. In a few cases, it might be impossible to make the matching correct, because transactions are being run concurrently. In most situations however, there are strategies that you can use to match up the two types of records with reasonable accuracy.

If the resources used in the individual transaction are the basis for accounting, then when you have matched up the CICS performance records and the Db2 accounting records, you can relate them back to the specific user. Alternatively, you can 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 user.

The two main factors that determine what strategies you should use are:

- Whether each CICS transaction ID represents only one possible transaction path (and so it always represents the same amount of resources consumed), or whether many different transaction paths share the same CICS transaction ID (so it can represent different amounts of resources consumed.
- Whether each Db2 accounting record relates to only one transaction or a part of one transaction (because you have taken one of the measures described in “Controlling the relationship between Db2 accounting records and CICS performance class records” on page 124), or whether it contains information about more than one transaction.

Figure 35 on page 126 shows how these factors combine to create four typical scenarios that you might encounter when matching Db2 accounting records and CICS performance class records. The following sections suggest strategies for matching the records in each case, and for charging the resources used back to the user.

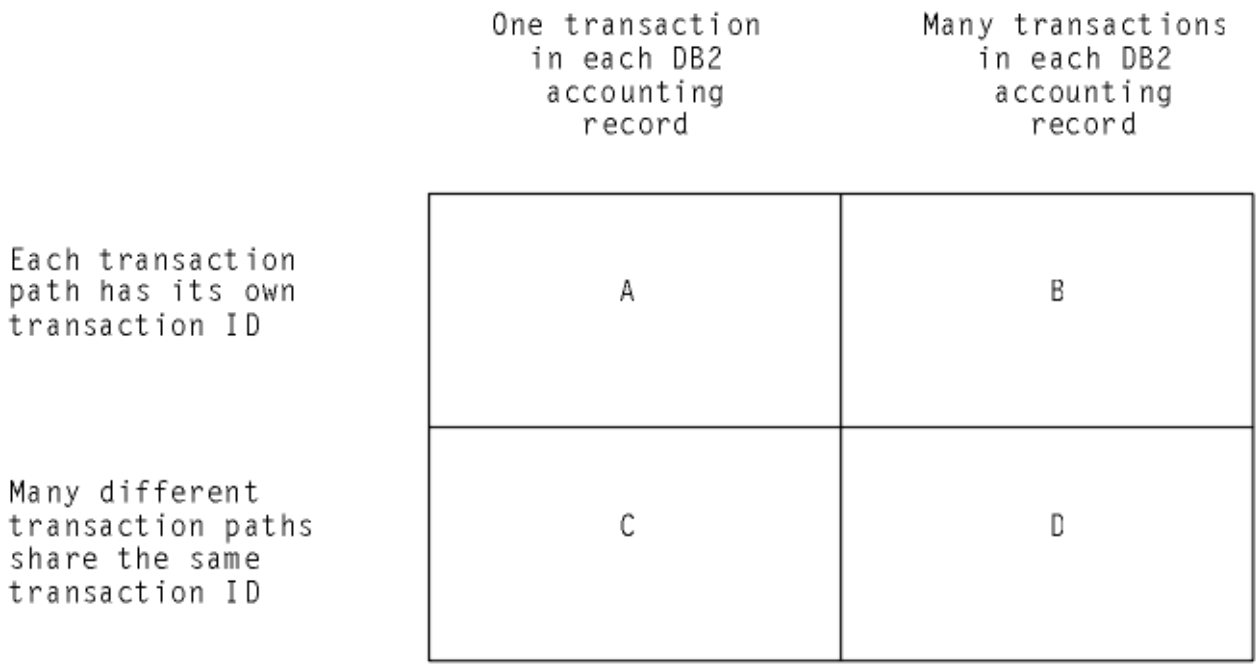


Figure 35. Different accounting scenarios

Scenario A: One transaction in each Db2 accounting record, with its own transaction ID

In this scenario, you know that each Db2 accounting record contains information relating to a single, identifiable CICS transaction. If the transaction accessed Db2 resources more than once, Db2 might have created more than one accounting record for it. You must match the Db2 accounting records relating to the transaction, to the CICS performance record for the transaction.

The user for the transaction can be identified from the CICS performance record. This record contains the CICS activities related to this transaction. You can identify the Db2 accounting records that apply to this transaction by using any of the data items listed in “Using data in the Db2 accounting record to identify the corresponding CICS performance class records” on page 125.

As all these transactions are identical, you can expect that they consume comparable amounts of resources. For accounting purposes, you could create model transactions for each transaction type. Because you can identify which Db2 accounting records apply to which CICS transactions, you can match up the Db2 accounting records and the CICS performance record for one transaction, and then assign the amount of Db2 resources used in those accounting records, to each subsequent transaction of that type. You should validate the correctness of your models on a regular basis, in case the resource usage changes.

Scenario B: Several transactions in each Db2 accounting record, but each transaction type has its own transaction ID

In this scenario, each Db2 accounting record can contain information relating to more than one transaction, so you cannot match each Db2 accounting record directly to the relevant CICS performance record. However, you can identify the types of transaction that are present in the Db2 accounting record, because each transaction ID only refers to one type of transaction.

If only one type of CICS transaction is present in a particular Db2 accounting record, then for accounting purposes, the resources consumed in Db2 can be split equally between each transaction. This is reasonable, because 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, as noted in “Controlling the relationship between Db2 accounting records and CICS performance class records” on page 124, units of work in the same transaction might use a different

thread, and so not be present in the same Db2 accounting record. Make sure that you have assigned all the relevant Db2 resources to each transaction; this can involve examining more than one Db2 accounting record.

If two or more different types of CICS transaction are present in a particular Db2 accounting record (because they use the same DB2ENTRY and hence the same thread), you cannot use the method of distributing the resources equally, because the different types of transaction might use different resources. In this case, you can create model transactions by periodically measuring the amount of Db2 resources used by each type of CICS transaction. Take these measurements by temporarily disallowing thread reuse, and looking at the resulting Db2 accounting records, which contains information relating to only one transaction. Use these model transactions to charge back the resources to the user. You should periodically validate the correctness of the model transactions.

Scenario C: One transaction in each Db2 accounting record, but several types of transaction use the same transaction ID

In this scenario, you know that each Db2 accounting record contains information relating to a single CICS transaction, but because several types of transaction use the same transaction ID, you cannot be sure which type of transaction is shown by a particular Db2 accounting record.

You cannot match up a set of records for one instance of a transaction and then reuse those figures, as you could in Scenario A. You must match all the individual CICS performance records with their corresponding Db2 accounting records. Unless you do this, you cannot know what type of transaction is represented by each Db2 record.

You can match each of the Db2 accounting records to the relevant CICS performance record by using the data items listed in “Using data in the Db2 accounting record to identify the corresponding CICS performance class records” on page 125. If you have specified either ACCOUNTREC(TASK) or ACCOUNTREC(UOW) in the DB2ENTRY or DB2CONN, so that CICS passes its LU6.2 token to Db2, then you can match the records together easily. If not, you must match the records based on their time stamps. In this case, the matching might not be accurate if transactions are being run simultaneously.

You can then use your matched sets of records to charge back the resources used for each transaction, to the user identified by the CICS performance record.

Scenario D: Several transactions in each Db2 accounting record, and several transaction types use the same transaction ID

In this scenario, each Db2 accounting record can contain information relating to more than one transaction, and also you cannot use the transaction IDs to identify which types of transaction are present in the accounting record.

This situation is best avoided, because you are unlikely to be able to match records accurately. If you do find yourself in this situation, the best solution is to create model transactions, as described for Scenario B. Next, find a way to mark the CICS performance records with an identifier that is unique to each transaction. For example, the user could supply information in a user field in the performance records that identifies the transaction being executed. Now you can use this field to identify which of the model transactions should be used for accounting in this case.

Accounting for processor usage in a CICS Db2 environment

Information about processor time consumption is provided in the Db2 accounting and statistics trace records. You can use these records to calculate the total processor time used in CICS and Db2.

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 Db2 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 collect package level accounting in Db2 and package level accounting wait in Db2. For information on package level accounting, see Managing Db2 performance in Db2 for z/OS product documentation.

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.

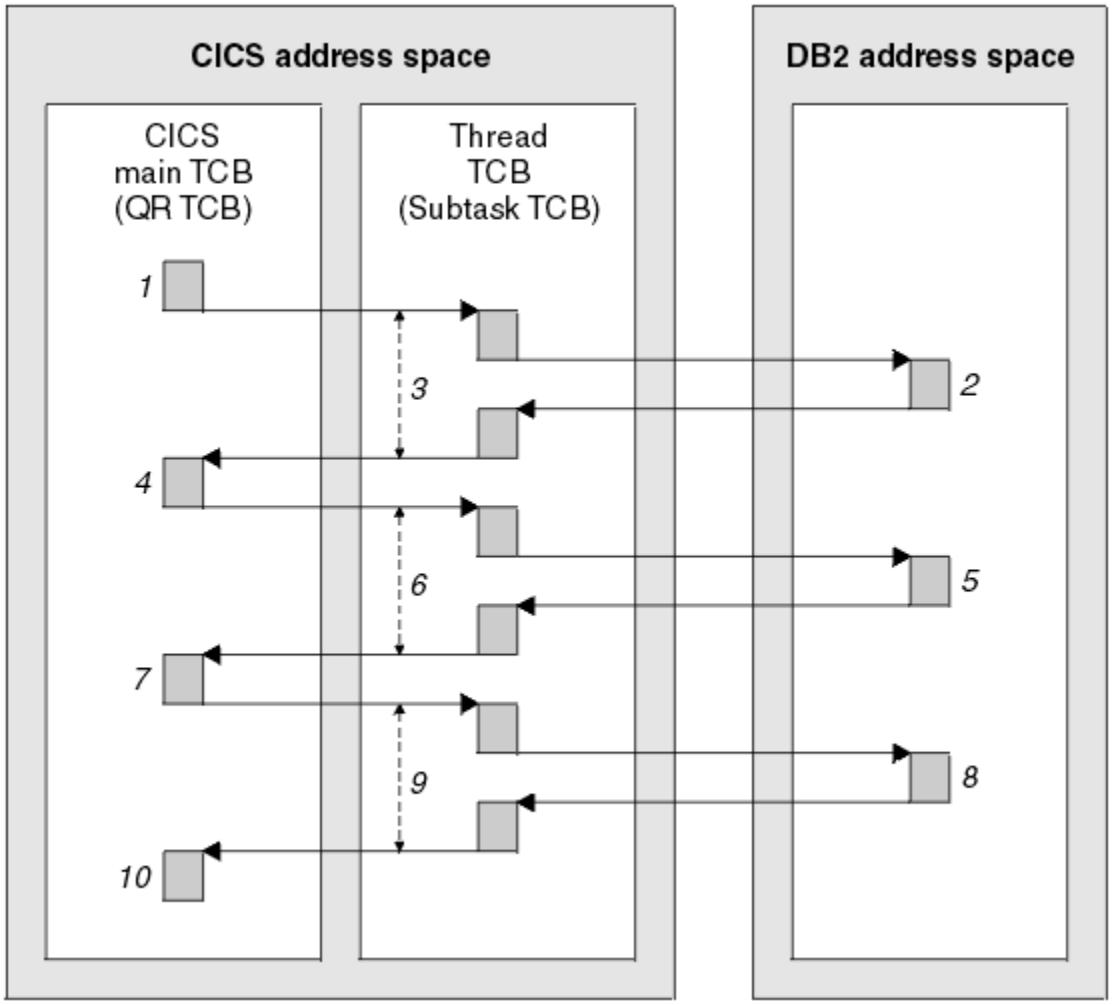
In a CICS Db2 environment, the processor time from the Db2 accounting records is typically much greater than the processor time reported in the Db2 statistical records, because most of the processor time used is in the thread TCB itself and in the Db2 address spaces using cross memory services.

Db2 accounting records are produced when a thread is terminated or signon occurs. This means that the period reported in the Db2 accounting record is the time between thread start or user signon (if reusing a thread previously used by another user) and thread termination or another signon. If several different transactions are specified for the same DB2ENTRY, and they use the same CICS transaction ID and the same Db2 authorization ID to sign on to the thread, then the Db2 accounting record for this thread can include data for more than one transaction. Also, if a transaction releases its Db2 thread at syncpoint, and the thread is terminated or re-used by another transaction, then the original transaction has to use a different thread. A single transaction can therefore be associated with multiple Db2 threads during the life of its execution. As Db2 produces accounting records for each thread, this can mean that multiple accounting records are produced for the transaction. See “Controlling the relationship between Db2 accounting records and CICS performance class records” on page 124 for information on how you can determine the transactions that are included in each Db2 accounting record.

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 UOW. The elapsed time for a thread is associated with the thread and not with the transaction.

Figure 36 on page 129 and Figure 37 on page 131 show each period of processor time that is reported by CICS and Db2, and where it takes place. The location of processing differs depending on whether or not the application accessing Db2 is threadsafe. This is because when CICS is connected to Db2, and is exploiting the open transaction environment, the CICS Db2 attachment facility uses CICS-managed open TCBs rather than CICS Db2 subtask TCBs.

Figure 36 on page 129 shows the situation when CICS is connected to Db2 and the application accessing Db2 is not threadsafe.




```
This picture shows the environment when CICS is connected
to Db2 when the application
accessing Db2 is not threadsafe.

Times 3 + 6 + 9 are reported as CLASS 1 processor time
(time spent under the thread TCB)
Times 2 + 5 + 8 are reported as CLASS 2 processor time
(time spent in DB2)
Times 1 + 4 + 7 + 10 are reported as CICS processor time
(time spent on the CICS main TCB)
```

Figure 36. CICS with Db2 and non-threadsafe application: Processor times recorded

If CICS is connected to Db2 but the application accessing Db2 is not threadsafe, then CICS is using an open TCB to run the thread into Db2, but it is using the open TCB in the same way as it would use a subtask TCB. The TCB that CICS uses to run the thread into Db2 is called the thread TCB.

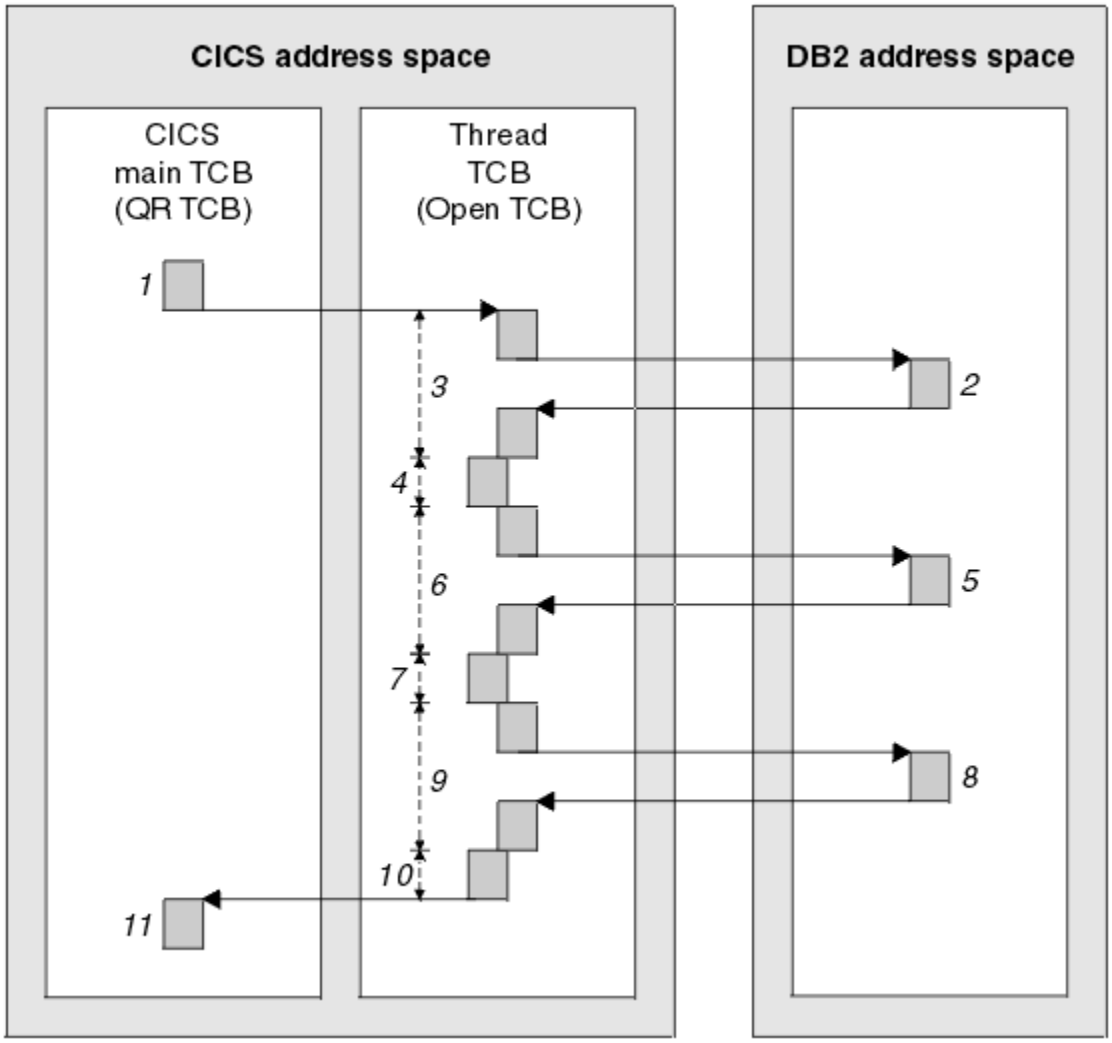
The periods of processor time shown in the figure are as follows:

- Time 1: The application starts on the CICS main TCB. At the end of Time 1, the application issues an EXEC SQL request.
- Time 2: Db2 is fulfilling the application's request. This processor time is spent in the Db2 address space. At the end of Time 2, Db2 passes its response to the CICS Db2 attachment facility.
- Time 3: The CICS Db2 attachment facility is carrying out processing on the thread TCB. This covers both the processing needed for the application to access Db2, and the processing needed to pass the Db2 response back to the application. It also includes the processor time taken for Db2 to fulfill the request

and respond. Time 2 is therefore nested within Time 3. At the end of Time 3, the response from Db2 is passed to the application, and the access to Db2 is complete.

- Time 4: The application has finished with Db2 for the moment. The application code runs on the CICS main TCB, until the application needs to access Db2 again. At the end of Time 4, the application issues a second EXEC SQL request.
- Time 5: As for Time 2, Db2 is fulfilling the second request from the application.
- Time 6: The CICS Db2 attachment facility is carrying out processing on the thread TCB related to this second request, or waiting for a response from Db2.
- Time 7: The application code runs again on the CICS main TCB. It issues a third EXEC SQL request.
- Time 8: As for Time 2, Db2 is fulfilling a third request from the application.
- Time 9: The CICS Db2 attachment facility is carrying out processing on the thread TCB related to this third request, or waiting for a response from Db2.
- Time 10: The application has now made all its EXEC SQL requests. The application code continues to run on the CICS main TCB until it terminates.

Figure 37 on page 131 shows the situation when CICS is connected to Db2 and the application accessing Db2 is threadsafe.




```
This picture shows the environment when CICS is connected to Db2,
and when the application accessing Db2 is threadsafe.

Times 3 + 4 + 6 + 7 + 9 + 10 are reported as CLASS 1 processor time
(time spent under the open TCB, processing access to DB2
and running application code)
Times 2 + 5 + 8 are reported as CLASS 2 processor time
(time spent in DB2)
Times 1 + 11 are reported as CICS processor time
(time spent on the CICS main TCB)
```

Figure 37. CICS with Db2 and threadsafe application: Processor times recorded

Here, CICS is using an open TCB to run the thread into Db2. As the application is threadsafe, the application code can also run on the open TCB. The periods of processor time shown in the figure are as follows:

- Time 1: The application starts on the CICS main TCB. At the end of Time 1, the application issues an EXEC SQL request.
- Time 2: Db2 is fulfilling the application's request. This processor time is spent in the Db2 address space. At the end of Time 2, Db2 passes its response to the CICS Db2 attachment facility.
- Time 3: The CICS Db2 attachment facility is carrying out processing on the thread TCB. This covers both the processing needed for the application to access Db2, and the processing needed to pass the response from Db2 back to the application. It also includes the processor time taken for Db2 to fulfil the

request and respond. Time 2 is therefore nested within Time 3. At the end of Time 3, the Db2 response is passed to the application, and the access to Db2 is complete.

- Time 4: The application has finished with Db2 for the moment. As the application is threadsafe, the application code now runs on the thread TCB (the open TCB). At the end of Time 4, the application issues a second EXEC SQL request.
- Time 5: As for Time 2, Db2 is fulfilling the second request from the application.
- Time 6: The CICS Db2 attachment facility is carrying out processing on the thread TCB related to this second request, or waiting for a response from Db2.
- Time 7: The application code runs again on the thread TCB. It issues a third EXEC SQL request.
- Time 8: As for Time 2, Db2 is fulfilling a third request from the application.
- Time 9: The CICS Db2 attachment facility is carrying out processing on the thread TCB related to this third request, or waiting for a response from Db2.
- Time 10: The application has now made all its EXEC SQL requests. The application code runs again on the thread TCB.
- Time 11: The application has completed its processing. The application terminates, and processing returns to the CICS main TCB.

Note that an application that is defined as threadsafe (that is, you have told CICS that the application code is threadsafe), can still issue CICS commands that are not threadsafe. These commands force a switch back to the CICS main TCB, and the application code then continues to run on the CICS main TCB until the application makes its next EXEC SQL request. The activity associated with the non-threadsafe CICS command, and with the application code that runs on the CICS main TCB, is reported as CICS processor time, and not as CLASS 1 processor time. Once the EXEC SQL request is issued, the application code can run on the open TCB again, and is reported from then on as CLASS 1 processor time.

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, the timer is checked again and both the timer start and end values are recorded in the SMF type 101 record.

In Figure 36 on page 129, CLASS 1 is the sum of the times 3, 6, and 9. In Figure 37 on page 131, CLASS 1 is the sum of the times 3, 4, 6, 7, 9, and 10.

The fields in the SMF type 101 record, the Db2 accounting 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 Managing Db2 performance in Db2 for z/OS product documentation. There is also the description of accounting record fields in member DSNWMSGS, which is shipped in SDSNSAMP.

When CLASS 1 recording becomes active for a thread, it is recording time spent on the L8 open TCB. Because the L8 TCB is used for both CICS activity and Db2 activity, this includes processor time spent in the CICS-Db2 attachment facility, including trace calls. It also includes processor time spent running application code (if the application is threadsafe) and threadsafe CICS commands on the open TCB. If a thread is reused, the thread housekeeping processor time is also included in the CLASS 1 processor time. As in previous releases, there is a proportion of thread creation and thread termination processing that is not captured by CLASS 1 time. The CLASS 1 processor time does not include any time spent running application code on the QR TCB, which happens as follows:

- In the initial period before the application issues its first EXEC SQL request and moves on to the open TCB (time 1 in the figure). This initial application code runs on the CICS main TCB, and is not seen by Db2 accounting.

- If the application issues a CICS command that is not threadsafe. This forces processing to move back to the CICS main TCB. The application code then continues to run on the CICS main TCB, where it cannot be seen by Db2 accounting, until the application issues its next EXEC SQL request. At this point, processing can move back onto the open TCB.

Unless you are using Db2 sysplex query parallelism (parallel query), you do not need to use the Db2 CLASS 1 processor time for accounting purposes. The processor time recorded in the CICS SMF type 110 record is all that is required to give a complete account of the processor time consumed by an application. This record includes the time spent in the application code, the thread creation and termination costs, and the time covered by the Db2 CLASS 1 processor time. For more information, see “Calculating CICS and Db2 processor times for Db2” on page 134.

Accounting CLASS 2 processor time

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 field in the SMF type 101 record used for accounting CLASS 2 processor time is QWACAJST.

In Figure 36 on page 129 and Figure 37 on page 131, CLASS 2 is the sum of the times 2, 5, and 8.

For a description of the contents of the Db2 statistics records, see Managing Db2 performance in Db2 for z/OS product documentation.

The elapsed time (start and end times between the defined points) is also reported in the SMF type 101 record (QWACASC).

When CICS is connected to Db2 there can be a significant difference between CLASS 1 and CLASS 2 processor time, as there is for IMS and TSO. This is because when CICS is connected to Db2, the CLASS 1 processor time includes processor time spent in the CICS-Db2 attachment facility, including trace calls, and also includes processor time spent running threadsafe application code and threadsafe CICS commands on the open TCB. All this processor time occurs in the CICS address space, and so is not reported for accounting CLASS 2. The CLASS 2 processor time itself is not affected by the open transaction environment.

Unlike accounting CLASS 1, the CLASS 2 elapsed times and processor times accurately reflect the elapsed times and processor times spent 'IN DB2'. Accounting CLASS 2 information is therefore very useful for monitoring the performance of SQL execution. However, the processor overhead associated with having class accounting is quite considerable.

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 IDs. The processor overhead is 0-5%, with 2% being typical.

Calculating CICS and Db2 processor times for DB2 Version 5 or earlier

When your CICS system is connected to DB2 Version 5 or earlier, the CICS performance class record does not include processor time consumed in Db2. To estimate the total processor time for a single transaction, add information from the corresponding CICS performance record and the Db2 accounting record (SMF type 101 record).

About this task

For information about matching CICS performance records with Db2 accounting records, see “Relating Db2 accounting records to CICS performance class records” on page 122.

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 Db2 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. Capture ratio is the ratio of reported CPU time to total used CPU time (for more information, see z/OS Resource Measurement Facility (RMF) User's Guide.

Calculating CICS and Db2 processor times for Db2

When CICS is connected to Db2 and is exploiting the open transaction environment, the CICS Db2 attachment facility uses CICS managed open TCBs rather than CICS Db2 subtask TCBs.

About this task

Using CICS managed open TCBs means the CICS monitoring facility can measure activity that was previously only reported in the Db2 accounting record (the SMF type 101 record). For example, CICS can now measure the processor time consumed on the Db2 thread and the processor time consumed in Db2 (the CLASS 1 and CLASS 2 CPU time). When CICS is using L8 open TCBs, the CPU time reported for these TCBs by the CICS monitoring facility includes the Db2 CLASS 1 processor time.

In the open transaction environment, the CICS L8 task processor time can also include the cost of creating a Db2 thread.

Unless you are using Db2 parallel query, **do not** add together the processor time from the CICS records (SMF type 110 records) and the Db2 accounting records (SMF type 101 records) when calculating the total processor time for a single transaction, because the Db2 processor time would then be included twice. The total processor time for a single transaction is recorded in the USRCPUT field in the CICS records (performance class data field 008 from group DFHTASK). This field includes all processor time used by the transaction when it was executing on any TCB managed by the CICS dispatcher. CICS-managed TCBs include the QR, RO, CO, and L8 mode TCBs.

If you are using Db2 parallel query, you do need to add some of the processor time from the Db2 accounting records. With Db2 sysplex query parallelism, Db2 might use multiple TCBs within the Db2 address space to service a query. The TCBs used for this purpose are non-CICS TCBs, so any processor time that they consume is not accounted for in the CICS SMF type 110 records. To obtain the total amount of processor time used for a parallel query, you therefore need to add the PAR.TASKS processor time recorded in the Db2 SMF type 101 record that applies to the transaction, to the processor time recorded in the USRCPUT field in the CICS SMF type 110 record.

Chapter 9. Troubleshooting Db2

This section discusses problem determination for CICS Db2.

These topics contain Diagnosis, Modification or Tuning information.

Thread TCBs (task control blocks)

In the CICS Db2 environment, each thread into Db2 runs under a thread task control block (TCB). This section provides further technical information about thread TCBs, to assist with problem determination.

See Overview: How threads work for an overview of thread TCBs.

Thread TCBs are open L8 mode TCBs. The open TCBs are “daughters” of the main CICS TCB (the QR TCB). The CICS Db2 task-related user exit itself runs on the open TCB, as well as using it to run the thread. The task-related user exit uses the CICS Db2 attach module DFHD2D2 to invoke Db2 when it needs to acquire a thread. Another module, DFHD2CO, running on a different TCB, deals with aspects of the overall CICS Db2 connection, including identifying to Db2 and disconnecting CICS from Db2.

The maximum number of open TCBs that can be running threads into Db2 at any one time is controlled using the TCBLIMIT parameter of the DB2CONN. An open TCB running a thread is not terminated when the thread is terminated. An open TCB can be terminated if:

- A CICS transaction is force purged from CICS and the thread is still active in Db2. In this case the TCB is terminated as a means of flushing the request out of Db2. The current UOW in Db2 is backed out.
- CICS is in doubt as to the outcome of a UOW because it has lost contact with its coordinator. Terminating the TCB 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 dispatcher, where open TCBs are returned if they are not being used by the CICS Db2 attachment facility, cleans up the unused open TCBs after 30 minutes.

Wait types for CICS Db2

The CICS Db2 task-related user exit, DFHD2EX1, issues **WAIT_MVS** calls for different purposes. These calls are distinguished by their resource name and type values. The resource name and resource type values are visible in the dispatcher section of a CICS system dump and also appear on the **CEMT INQUIRE TASK** panel as Hty and Hva values respectively.

The CICS Db2 task-related user exit and the request into Db2 run on an L8 open TCB. The CICS task is not put into a CICS dispatcher wait when active in Db2, so there is no related information in the dispatcher section of the CICS system dump. The **CEMT INQUIRE TASK** panel shows that the CICS Db2 task is running on an open TCB. To find out if the task is active in Db2, use the **DSNC DISPLAY TRAN** command. This command displays all the threads currently in use, and the task with which each thread is associated. See DISPLAY PLAN or TRAN output for an example of the output from the command. If the thread associated with the task has a status of '*' in the 'S' field, this shows that the thread is currently active in Db2. The task is therefore either running or waiting in Db2.

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 11 on page 136. A fuller explanation of each wait follows the table.

Table 11. 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 task to complete the request. Used when CICS is connected to DB2 Version 5 or earlier.	Purge: No. Forcepurge: Yes
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
CDB2CONN		The CICS task has an open TCB but is waiting for a Db2 connection to become available to use with the open TCB. This indicates that TCBLIMIT has been reached, and the maximum permitted number of open TCBs (and hence connections) are being used to access Db2.	No

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. If you attempt to forcepurge the task, message DFHAP0604 is issued to the console, and forcepurge is deferred until a thread has been acquired and the task is no longer queued waiting for a thread. Instead of purging the task, you can use a SET DB2ENTRY() THREADLIMIT(*n*) command 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.

CDB2CONN indicates that the CICS task has obtained an open TCB from the pool of L8 and L9 mode open TCBs, but is waiting for a Db2 connection to become available to use with the open TCB. This shows that the TCBLIMIT specified in the DB2CONN has been reached, which limits the number of open TCBs (and hence connections) that can be used to access Db2. When the TCBLIMIT is reached, the CICS Db2 task-related user exit can obtain a TCB from the pool of L8 and L9 mode open TCBs, but it must wait before it can use the TCB to run a thread into Db2. When another task stops using its L8 or L9 mode TCB to run a thread into Db2, and so the number of TCBs in use running threads falls below TCBLIMIT, the waiting task is allowed to use its own L8 or L9 mode TCB to run a thread into Db2.

Purge of the task is not allowed for a CDB2CONN wait. If you attempt to forcepurge the task, message DFHAP0604 is issued to the console, and forcepurge is deferred until a Db2 connection has been acquired. Instead of purging the task, you can use a SET DB2CONN TCBLIMIT command to increase TCBLIMIT, which increases the number of open TCBs permitted to access Db2. If you increase TCBLIMIT, CICS posts tasks to try again to acquire a Db2 connection.

If the limit set automatically by CICS for the number of L8 and L9 mode open TCBs is reached, so no more open TCBs can be created, the task is suspended with HTYPE(DISPATCH) and HVALUE(OPEN_TCB). CICS sets this limit using the formula `(2 * MXT value) + 32`, with the MXT or MAXTASKS value for the CICS region. The task is suspended with HTYPE(CDB2CONN) in the situation where this limit is not exceeded, but TCBLIMIT is exceeded.

Table 12 on page 137 gives details of WAITS issued using WAIT_OLDC dispatcher calls where the ECB is hand posted:

Table 12. 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 13 on page 137 shows details of EXEC CICS WAIT EXTERNAL requests issued by the CICS Db2 attachment facility.

Table 13. 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 14 on page 137 gives details of EXEC CICS WAIT EVENT requests issued by the CICS Db2 attachment facility:

Table 14. 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.
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 for CICS Db2

Messages issued by the CICS Db2 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 CICS messages and are available using the CMAC CICS-supplied transaction.

Trace for CICS Db2

The CICS Db2 attachment facility uses AP domain trace points in the range 3100 to 33FF.

The contents of all trace points issued by the CICS Db2 attachment facility are documented in CICS Db2 trace points.

Standard tracing performed by the CICS Db2 attachment facility is controlled by the RA (Resource Manager Adapter) and RI (RMI) trace flags. RMI trace controls:

- Trace output from the CICS Db2 task-related user exit, DFHD2EX1
- Trace output from the CICS Db2 thread processor DFHD2D2

All other standard tracing from the attachment is controlled using Resource Manager Adapter tracing. A large proportion of the possible trace issued from the CICS Db2 attachment facility is exception tracing, which is output irrespective of RI or RA tracing being active, but is issued independently. You can set the levels of RA and RI trace required by using the CICS-supplied transaction CETR, or by setting system initialization parameters **STNTRRA** and **STNTRRI**.

CICS Db2 trace output is written to the CICS internal trace table and auxiliary trace and GTF trace destinations if active.

Sample trace output from the RMI and the CICS Db2 TRUE

This is a sample 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. In this example the CICS Db2 task-related user exit is running on an open L8 mode TCB, and uses that same TCB to run threads into Db2.

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

              TASK-00035 KE_NUM-003F TCB-QR   /007C5B60 RET-99902F92 TIME-08:32:00.7673270795
INTERVAL-00.0000135312        =000162=
                1-0000  01                              *.                              *
                2-0000  C4E2D5C3 E2D8D340               *DSNCSQL                        *
                3-0000  B60AF030 A54A8EC0                                                       *..0.v$.
{                        *
                4-0000  E3C5E2E3 D7F0F540 00000001 00100000  19900000 999000E0 000114E0 00000000
*TESTP05 ............r..\...\....*
                0020  00000020 00007BB0 00000000 98400000  0040000A 00101100 00000000 00000000
*......#.....q ... ..............*
                0040  00000000 00000000 00000000
*............                    *

DS 0002 DSAT  ENTRY - FUNCTION(CHANGE_MODE) MODENAME(L8)

              TASK-00035 KE_NUM-003F TCB-QR   /007C5B60 RET-8009E318 TIME-08:32:00.7673341108
INTERVAL-00.0000070312        =000163=
                1-0000  00980000 00000003 00000001 00000000  A0000040 00000000 05000102 01000002
*.q................. ............*
                0020  01007450 00000000 000002B0 18CCB490  18CCB6D8 9801014C 01031498 18220F30
*...&.............Qq..<...q....*
                0040  18CCBB70 1833E000 9800E8F0 00007000  190441EC 18CCBB70 00000008 D3F8D8D9
*......\.q.Y0.................L8QR*
                0060  18CCB638 FFFFFFFF 18CB4B00 987D9520  00000001 18DEA4A0 18DEA010 00000001
*............q'n.......u.........*
                0080  00007550 18CCB776 00000000 0000E8F0  18CCBA28 18CCBA28
*...&..........Y0........        *

DS 0003 DSAT  EXIT - FUNCTION(CHANGE_MODE) RESPONSE(OK)

              TASK-00035 KE_NUM-003F TCB-L8000/007C6B90 RET-8009E318 TIME-08:32:00.7673710483
INTERVAL-00.0000369375        =000164=
                1-0000  00980000 00000003 00000001 00000000  A0000040 00000000 05000102 01000002
*.q................. ............*
```

```
                0020  01007450 00000000 000002B0 18CCB490  18CCB6D8 9801014C 01031498 18220F30
*...&.............Qq..<...q....*
                0040  18CCBB70 1833E000 9800E8F0 00007000  190441EC 18CCBB70 00000001 D3F8D8D9
*......\.q.Y0.............L8QR*
                0060  18CCB638 FFFFFFFF 18CB4B00 987D9520  00000001 18DEA4A0 18DEA010 00000001
*...........q'n.......u.........*
                0080  00007550 18CCB776 00000000 0000E8F0  18CCBA28 18CCBA28
*...&.........Y0........        *

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

             TASK-00035 KE_NUM-003F TCB-L8000/007C6B90 RET-99902F92 TIME-08:32:00.7673729077
INTERVAL-00.0000018593        =000165=
               1-0000  01
*.                            *
               2-0000  C4E2D5C3 E2D8D340
*DSNCSQL                      *
               3-0000  B60AF030 A54A8EC0                                                  *..0.v$.
{                             *
               4-0000  18CCB654 0006F004 19044204 00000000  007A8000 007A8000 18F0005C 18F0CCD0
*......0.........:...:...0.*.0.}*
                0020  190441C0 19044210 19044206 18F000D0  19044200 18CCB490 190441C8 190441B0  *...
{.........0.}...........H....*
                0040  190441AC 190441EC 18CCB5F3 190441FD  18CCB5F4 190441B4 190441B2 00020000
*...........3.......4...........*
               5-0000  D3F8
*L8                           *

AP 3180 D2EX1 ENTRY - APPLICATION REQUEST - EXEC SQL FETCH

             TASK-00035 KE_NUM-003F TCB-L8000/007C6B90 RET-8009E4D0 TIME-08:32:00.7673780483
INTERVAL-00.0000051406        =000166=
               1-0000  02000000 00280800 001EE3C5 E2E3D7F0  F5401663 1BA501CD D1DC0002 18F0BA40
*..........TESTP05 ...v..J....0. *
                0020  00000000 18F0CDB0 018D0004
*.....0......                 *
               2-0000  00DE6EC4 C6C8C4F2 D3D6E340 40404040  E7D7F0F5 184AC080 18DF2D78 18DCD030  *..>DFHD2LOT
XP05.${.......}.*
                0020  00000000 0006EFF0 00000000 18DF2E0C  18F0CDE0 00000000 00000000 FF6CD800
*.......0.........0.\.........%Q.*
                0040  00000000 00000000 00000000 00000000  00000000 E3C5E2E3 D7F0F540 000C010C
*...................TESTP05 ....*
                0060  A000C000 00000000 00000000 00000000  C9E8D2F2 E9F2C7F1 B60AF030 A54A8EC0  *..
{.............IYK2Z2G1..0.v$.{*
                0080  D1E3C9D3 D3C9F140 40404040 40404040  00000000 00000000 C7C2C9C2 D4C9E8C1
*JTILLI1         ........GBIBMIYA*
                00A0  C9E8C1D8 E3C3F0F3 0AF030A5 4A8EC6D9  C2400003 000118F0 CD680000 00000000
*IYAQTC03.0.v$.FRB .....0........*
                00C0  00000000 00000000 39050001 04000000  00000000 00000000 00000000 0000
*............................  *
```

```
AP 3250 D2D2 ENTRY - FUNCTION(DB2_API_CALL) CSUB_TOKEN(18DCD030)

             TASK-00035 KE_NUM-003F TCB-L8000/007C6B90 RET-98E3321C TIME-08:32:00.7674652827
INTERVAL-00.0000872343        =000167=
               1-0000  00200000 00000034 00000000 00000000  B8000000 00000000 03000100 18DCD030
*................................}.*
               2-0000  03006EC4 C6C8C4F2 C3E2C240 40404040  B60AF031 FAD0CB43 18D305E0 18DF2D78
*..>DFHD2CSB     ..0..}...L.\....*
                0020  19044210 007C6B90 00000000 00000000  B60AF030 A54A8EC0 00000000 00000000
*.....@,...........0.v$.{........*
                0040  18DF2DE4 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*...U............................*
                0060  00000000 00000000 00000000 E3C5E2E3  D7F0F540 D1E3C9D3 D3C9F140 40404040  *...........TESTP05
JTILLI1     *
                0080  40404040 C5D5E3D9 E7D7F0F5 F0F0F0F1  00000000 B60AF032 01359743 C7C2C9C2  *
ENTRXP050001......0...p.GBIB*
                00A0  D4C9E8C1 C9E8C1D8 E3C3F0F3 0AF030A5  4A8E0000 44800000 00000001 00000000
*MIYAIYAQTC03.0.v$..............*
                00C0  00000000 00000000 00000000 40404040  40404040 40404040 40404040 FF6CD800
*...........                  .%Q.*
                00E0  C6D9C240 00030001 18F0CD68 00000000  00000000 00000000 00003905 00010400
*FRB .....0.....................*
                0100  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                0120  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                0140  00000000 00000000 00000000 00000000  00000000 00000000 98DCD110 00000000
*....................q.J.....*
                0160  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                0180  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                01A0  00000000 00000000 40404040 40404040  40404040 40404040 40404040 40404040
*........                      *
                01C0  40404040 40404040 40404040 40404040  40404040 40404040 40404040 00000000
*                        ....*
                01E0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                0200  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
```

```
                  0220  00000000 00000000 00000000 00000000  00000000 00000000 00000004 18DCD2B0
*.............................K.*
                  0240  6E6EE399 81838540 E2A38199 A3406E6E  0100035C C9C4C5D5 00000000 00000000  *>>Trace Start
>>...*IDEN........*
                  0260  0200035C E2C9C7D5 00000000 00000000  0300035C C3E3C8C4 00000000 00000000
*...*SIGN..........*CTHD........*
                  0280  0400035C C1D7C940 00000000 00000000  00000000 00000000 00000000 00000000
*...*API .........................*
                  02A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  02C0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  02E0  00000000 00000000 00000000 00000000  4C4CE399 81838540 C5958440 40404C4C
*................<<Trace End   <<*
                3-0000  00DE6EC4 C6C8C4F2 D3D6E340 40404040  E7D7F0F5 184AC080 18DF2D78 18DCD030  *..>DFHD2LOT
XP05.${.......}.*
                  0020  00000000 0006EFF0 00000000 18DF2E0C  18F0CDE0 00000000 00000000 FF6CD800
*.......0.........0.\.........%Q.*
                  0040  00000000 00000000 00000000 00000000  00000000 E3C5E2E3 D7F0F540 010C010C
*...................TESTP05 ....*
                  0060  A000C000 00000000 00000000 00000000  C9E8D2F2 E9F2C7F1 B60AF030 A54A8EC0  *..
{.............IYK2Z2G1..0.v$.{*
                  0080  D1E3C9D3 D3C9F140 40404040 40404040  00000000 00000000 C7C2C9C2 D4C9E8C1
*JTILLI1         ........GBIBMIYA*
                  00A0  C9E8C1D8 E3C3F0F3 0AF030A5 4A8EC6D9  C2400003 000118F0 CD680000 00000000
*IYAQTC03.0.v$.FRB .....0........*
                  00C0  00000000 00000000 39050001 04000000  00000000 00000000 00000000 0000
*............................  *

AP 326C D2D2 EVENT - ABOUT_TO_ISSUE_DB2_API_REQUEST

              TASK-00035 KE_NUM-003F TCB-L8000/007C6B90 RET-98E3321C TIME-08:32:00.7674700952
INTERVAL-00.0000048125        =000168=
                1-0000  98DCD110 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*q.J.............................*
                2-0000  C6D9C240 00030001 18F0CDE0 00000000  00000000 00000000 00003905 00010400
*FRB .....0.\...................*
                  0020  00000000 00000000 00000000 00000000
*................                *
                3-0000  E3C5E2E3 D7F0F540
*TESTP05                        *
                4-0000  03006EC4 C6C8C4F2 C3E2C240 40404040  B60AF031 FAD0CB43 18D305E0 18DF2D78
*..>DFHD2CSB     ..0..}...L.\....*
                  0020  19044210 007C6B90 00000000 00000000  B60AF030 A54A8EC0 00000000 00000000
*.....@,...........0.v$.{........*
                  0040  18DF2DE4 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*...U............................*
                  0060  00000000 00000000 00000000 E3C5E2E3  D7F0F540 D1E3C9D3 D3C9F140 40404040  *............TESTP05
JTILLI1     *
                  0080  40404040 C5D5E3D9 E7D7F0F5 F0F0F0F1  00000000 B60AF032 01359743 C7C2C9C2  *
ENTRXP050001......0...p.GBIB*
                  00A0  D4C9E8C1 C9E8C1D8 E3C3F0F3 0AF030A5  4A8E0000 44800000 00000001 00000000
*MIYAIYAQTC03.0.v$...............*
                  00C0  00000000 00000000 00000000 40404040  40404040 40404040 40404040 FF6CD800
*............               .%Q.*
                  00E0  C6D9C240 00030001 18F0CDE0 00000000  00000000 00000000 00003905 00010400
*FRB .....0.\...................*
                  0100  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0120  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0140  00000000 00000000 00000000 00000000  00000000 00000000 98DCD110 00000000
*.....................q.J.....*
                  0160  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0180  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  01A0  00000000 00000000 40404040 40404040  40404040 40404040 40404040 40404040
*........                        *
                  01C0  40404040 40404040 40404040 40404040  40404040 40404040 40404040 00000000
*                     ....*
                  01E0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0200  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0220  00000000 00000000 00000000 00000000  00000000 00000000 00000004 18DCD2B0
*.............................K.*
                  0240  6E6EE399 81838540 E2A38199 A3406E6E  0100035C C9C4C5D5 00000000 00000000  *>>Trace Start
>>...*IDEN........*
                  0260  0200035C E2C9C7D5 00000000 00000000  0300035C C3E3C8C4 00000000 00000000
*...*SIGN..........*CTHD........*
                  0280  0400035C C1D7C940 00000000 00000000  00000000 00000000 00000000 00000000
*...*API .........................*
                  02A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  02C0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  02E0  00000000 00000000 00000000 00000000  4C4CE399 81838540 C5958440 40404C4C
*................<<Trace End   <<*
                5-0000  00DE6EC4 C6C8C4F2 D3D6E340 40404040  E7D7F0F5 184AC080 18DF2D78 18DCD030  *..>DFHD2LOT
XP05.${.......}.*
                  0020  00000000 0006EFF0 00000000 18DF2E0C  18F0CDE0 00000000 00000000 FF6CD800
*.......0.........0.\.........%Q.*
```

```
                  0040  00000000 00000000 00000000 00000000  00000000 E3C5E2E3 D7F0F540 010C010C
*...................TESTP05 ....*
                  0060  A000C000 00000000 00000000 00000000  C9E8D2F2 E9F2C7F1 B60AF030 A54A8EC0  *..
{.............IYK2Z2G1..0.v$.{*
                  0080  D1E3C9D3 D3C9F140 40404040 40404040  00000000 00000000 C7C2C9C2 D4C9E8C1
*JTILLI1         ........GBIBMIYA*
                  00A0  C9E8C1D8 E3C3F0F3 0AF030A5 4A8EC6D9  C2400003 000118F0 CD680000 00000000
*IYAQTC03.0.v$.FRB .....0........*
                  00C0  00000000 00000000 39050001 04000000  00000000 00000000 00000000 0000
*.............................  *
```

```
AP 326D D2D2 EVENT - RETURN_FROM_DB2_API_REQUEST

              TASK-00035 KE_NUM-003F TCB-L8000/007C6B90 RET-98E3321C TIME-08:32:00.7678738139
INTERVAL-00.0004037187        =000169=
                1-0000  98DCD110 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*q.J.............................*
                2-0000  C6D9C240 00030001 18F0CDE0 00000000  00000000 00000000 00003905 00010400
*FRB .....0.\....................*
                  0020  00000000 00000000 00000000 00000000
*................                *
                3-0000  E3C5E2E3 D7F0F540
*TESTP05                        *
                4-0000  03006EC4 C6C8C4F2 C3E2C240 40404040  B60AF031 FAD0CB43 18D305E0 18DF2D78
*..>DFHD2CSB     ..0..}...L.\....*
                  0020  19044210 007C6B90 00000000 00000000  B60AF030 A54A8EC0 00000000 00000000
*.....@,...........0.v$.{........*
                  0040  18DF2DE4 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*...U............................*
                  0060  00000000 00000000 00000000 E3C5E2E3  D7F0F540 D1E3C9D3 D3C9F140 40404040  *............TESTP05
JTILLI1     *
                  0080  40404040 C5D5E3D9 E7D7F0F5 F0F0F0F1  00000000 B60AF032 01359743 C7C2C9C2  *
ENTRXP050001......0...p.GBIB*
                  00A0  D4C9E8C1 C9E8C1D8 E3C3F0F3 0AF030A5  4A8E0000 44800000 00000001 00000000
*MIYAIYAQTC03.0.v$...............*
                  00C0  00000000 00000000 00000000 40404040  40404040 40404040 40404040 FF6CD800
*............                  .%Q.*
                  00E0  C6D9C240 00030001 18F0CDE0 00000000  00000000 00000000 00003905 00010400
*FRB .....0.\....................*
                  0100  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0120  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0140  00000000 00000000 00000000 00000000  00000000 00000000 98DCD110 00000000
*.......................q.J.....*
                  0160  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0180  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  01A0  00000000 00000000 40404040 40404040  40404040 40404040 40404040 40404040
*........                        *
                  01C0  40404040 40404040 40404040 40404040  40404040 40404040 40404040 00000000
*                        ....*
                  01E0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0200  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0220  00000000 00000000 00000000 00000000  00000000 00000000 00000005 18DCD2C0
*............................K{*
                  0240  6E6EE399 81838540 E2A38199 A3406E6E  0100035C C9C4C5D5 00000000 00000000  *>>Trace Start
>>...*IDEN........*
                  0260  0200035C E2C9C7D5 00000000 00000000  0300035C C3E3C8C4 00000000 00000000
*...*SIGN..........*CTHD........*
                  0280  0400035C C1D7C940 00000000 00000000  0500035C C1D7C940 00000000 00000000
*...*API ...........*API ........*
                  02A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  02C0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  02E0  00000000 00000000 00000000 00000000  4C4CE399 81838540 C5958440 40404C4C
*................<<Trace End   <<*

               5-0000  00DE6EC4 C6C8C4F2 D3D6E340 40404040  E7D7F0F5 184AC080 18DF2D78 18DCD030  *..>DFHD2LOT
XP05.${.......}.*
                  0020  00000000 0006EFF0 00000000 18DF2E0C  18F0CDE0 00000000 00000000 FF6CD800
*.......0.........0.\.........%Q.*
                  0040  00000000 00000000 00000000 00000000  00000000 E3C5E2E3 D7F0F540 010C010C
*...................TESTP05 ....*
                  0060  A000C000 00000000 00000000 00000000  C9E8D2F2 E9F2C7F1 B60AF030 A54A8EC0  *..
{.............IYK2Z2G1..0.v$.{*
                  0080  D1E3C9D3 D3C9F140 40404040 40404040  00000000 00000000 C7C2C9C2 D4C9E8C1
*JTILLI1         ........GBIBMIYA*
                  00A0  C9E8C1D8 E3C3F0F3 0AF030A5 4A8EC6D9  C2400003 000118F0 CD680000 00000000
*IYAQTC03.0.v$.FRB .....0........*
                  00C0  00000000 00000000 39050001 04000000  00000000 00000000 00000000 0000
*.............................  *

AP 3251 D2D2 EXIT - FUNCTION(DB2_API_CALL) RESPONSE(OK)

              TASK-00035 KE_NUM-003F TCB-L8000/007C6B90 RET-98E3321C TIME-08:32:00.7697274233
INTERVAL-00.0018536093        =000170=
```

```
               1-0000  00200000 00000034 00000000 00000000  B8000000 00000000 03000100 18DCD030
*.............................}.*
               2-0000  03006EC4 C6C8C4F2 C3E2C240 40404040  B60AF031 FAD0CB43 18D305E0 18DF2D78
*..>DFHD2CSB     ..0..}...L.\....*
                0020  19044210 007C6B90 00000000 00000000  B60AF030 A54A8EC0 00000000 00000000
*.....@,...........0.v$.{........*
                0040  18DF2DE4 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*...U............................*
                0060  00000000 00000000 00000000 E3C5E2E3  D7F0F540 D1E3C9D3 D3C9F140 40404040  *............TESTP05
JTILLI1     *
                0080  40404040 C5D5E3D9 E7D7F0F5 F0F0F0F1  00000000 B60AF032 01359743 C7C2C9C2  *
ENTRXP050001......0...p.GBIB*
                00A0  D4C9E8C1 C9E8C1D8 E3C3F0F3 0AF030A5  4A8E0000 44800000 00000001 00000000
*MIYAIYAQTC03.0.v$...............*
                00C0  00000000 00000000 00000000 40404040  40404040 40404040 40404040 FF6CD800
*............                .%Q.*
                00E0  C6D9C240 00030001 18F0CDE0 00000000  00000000 00000000 00003905 00010400
*FRB .....0.\....................*
                0100  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                0120  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                0140  00000000 00000000 00000000 00000000  00000000 00000000 98DCD110 00000000
*.......................q.J.....*
                0160  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                0180  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                01A0  00000000 00000000 40404040 40404040  40404040 40404040 40404040 40404040
*........                        *
                01C0  40404040 40404040 40404040 40404040  40404040 40404040 40404040 00000000
*                       ....*
                01E0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                0200  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                0220  00000000 00000000 00000000 00000000  00000000 00000000 00000005 18DCD2C0
*..............................K{*
                0240  6E6EE399 81838540 E2A38199 A3406E6E  0100035C C9C4C5D5 00000000 00000000  *>>Trace Start
>>...*IDEN........*
                0260  0200035C E2C9C7D5 00000000 00000000  0300035C C3E3C8C4 00000000 00000000
*...*SIGN..........*CTHD........*
                0280  0400035C C1D7C940 00000000 00000000  0500035C C1D7C940 00000000 00000000
*...*API ...........*API ........*
                02A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                02C0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                02E0  00000000 00000000 00000000 00000000  4C4CE399 81838540 C5958440 40404C4C
*................<<Trace End   <<*
               3-0000  00DE6EC4 C6C8C4F2 D3D6E340 40404040  E7D7F0F5 184AC080 18DF2D78 18DCD030  *..>DFHD2LOT
XP05.${.......}.*
                0020  00000000 0006EFF0 00000000 18DF2E0C  18F0CDE0 00000000 00000000 FF6CD800
*.......0.........0.\.........%Q.*
                0040  00000000 00000000 00000000 00000000  00000000 E3C5E2E3 D7F0F540 010C010C
*...................TESTP05 ....*
                0060  A000C000 00000000 00000000 00000000  C9E8D2F2 E9F2C7F1 B60AF030 A54A8EC0  *..
{.............IYK2Z2G1..0.v$.{*
                0080  D1E3C9D3 D3C9F140 40404040 40404040  00000000 00000000 C7C2C9C2 D4C9E8C1
*JTILLI1         ........GBIBMIYA*
                00A0  C9E8C1D8 E3C3F0F3 0AF030A5 4A8EC6D9  C2400003 000118F0 CDE00000 00000000
*IYAQTC03.0.v$.FRB .....0.\......*
                00C0  00000000 00000000 39050001 04000000  00000000 00000000 00000000 0000
*.............................  *
```

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

             TASK-00035 KE_NUM-003F TCB-L8000/007C6B90 RET-8009E4D0 TIME-08:32:00.7697348608
INTERVAL-00.0000074375        =000171=
               1-0000  02000000 00280800 001EE3C5 E2E3D7F0  F5401663 1BA501CD D1DC0002 18F0BA40
*.........TESTP05 ...v..J....0. *
                0020  00000000 18F0CDB0 018D0004
*.....0......                   *
               2-0000  00DE6EC4 C6C8C4F2 D3D6E340 40404040  E7D7F0F5 184AC080 18DF2D78 18DCD030  *..>DFHD2LOT
XP05.${.......}.*
                0020  00000000 0006EFF0 00000000 18DF2E0C  18F0CDE0 00000000 00000000 FF6CD800
*.......0.........0.\.........%Q.*
                0040  00000000 00000000 00000000 00000000  00000000 E3C5E2E3 D7F0F540 010C010C
*...................TESTP05 ....*
                0060  A000C000 00000000 00000000 00000000  C9E8D2F2 E9F2C7F1 B60AF030 A54A8EC0  *..
{.............IYK2Z2G1..0.v$.{*
                0080  D1E3C9D3 D3C9F140 40404040 40404040  00000000 00000000 C7C2C9C2 D4C9E8C1
*JTILLI1         ........GBIBMIYA*
                00A0  C9E8C1D8 E3C3F0F3 0AF030A5 4A8EC6D9  C2400003 000118F0 CDE00000 00000000
*IYAQTC03.0.v$.FRB .....0.\......*
                00C0  00000000 00000000 39050001 04000000  00000000 00000000 00000000 0000
*.............................  *
               3-0000  00
*.                              *
               4-0000  E2D8D3C3 C1404040 00000088 00000000  00004040 40404040 40404040 40404040
*SQLCA   ...h......             *
```

```
                  0020  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040
*                       *
                  0040  40404040 40404040 40404040 40404040  40404040 40404040 C4E2D540 40404040
*                        DSN     *
                  0060  00000000 00000000 00000000 FFFFFFFF  00000000 00000000 40404040 40404040
*.......................         *
                  0080  404040F0 F0F0F0F0                                                  *
00000                        *
                5-0000  00C86EC4 C6C8C4F2 C5D5E340 40404040  E7D7F0F5 40404040 B60AF017 63D17AC6  *.H>DFHD2ENT
XP05    ..0..J:F*
                  0020  E3C5E2E3 D7F0F540 00000000 00000000  00000000 00000000 D1E3C9D3 D3C9F140
*TESTP05 ................JTILLI1 *
                  0040  00808080 80000000 67E939C4 B0188B86  B60AF017 63D17AC6 0000000C 00000003
*.........Z.D...f..0..J:F........*
                  0060  00000001 00000001 00000000 00000000  00000001 00000001 00000000 00000000
*................................*
                  0080  00000000 00000002 00000001 00000000  00000000 00000000 00000000 00000000
*................................*
                  00A0  00000000 00000000 00000000 00000000  18DCD030 00000000 00000000 19044210
*..................}.............*
                  00C0  00000000 00000000
*........               *
                6-0000  03006EC4 C6C8C4F2 C3E2C240 40404040  B60AF031 FAD0CB43 18D305E0 18DF2D78
*..>DFHD2CSB     ..0..}...L.\....*
                  0020  19044210 007C6B90 00000000 00000000  B60AF030 A54A8EC0 00000000 00000000
*.....@,...........0.v$.{........*
                  0040  18DF2DE4 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*...U............................*
                  0060  00000000 00000000 00000000 E3C5E2E3  D7F0F540 D1E3C9D3 D3C9F140 40404040  *............TESTP05
JTILLI1     *
                  0080  40404040 C5D5E3D9 E7D7F0F5 F0F0F0F1  00000000 B60AF032 01359743 C7C2C9C2  *
ENTRXP050001......0...p.GBIB*
                  00A0  D4C9E8C1 C9E8C1D8 E3C3F0F3 0AF030A5  4A8E0000 44800000 00000001 00000000
*MIYAIYAQTC03.0.v$...............*
                  00C0  00000000 00000000 00000000 40404040  40404040 40404040 40404040 FF6CD800
*............                .%Q.*
                  00E0  C6D9C240 00030001 18F0CDE0 00000000  00000000 00000000 00003905 00010400
*FRB .....0.\....................*
                  0100  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0120  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0140  00000000 00000000 00000000 00000000  00000000 00000000 98DCD110 00000000
*.......................q.J.....*
                  0160  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0180  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  01A0  00000000 00000000 40404040 40404040  40404040 40404040 40404040 40404040
*........               *
                  01C0  40404040 40404040 40404040 40404040  40404040 40404040 40404040 00000000
*                         ...*
                  01E0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0200  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  0220  00000000 00000000 00000000 00000000  00000000 00000000 00000005 18DCD2C0
*..............................K{*
                  0240  6E6EE399 81838540 E2A38199 A3406E6E  0100035C C9C4C5D5 00000000 00000000  *>>Trace Start
>>...*IDEN........*
                  0260  0200035C E2C9C7D5 00000000 00000000  0300035C C3E3C8C4 00000000 00000000
*...*SIGN...........*CTHD........*
                  0280  0400035C C1D7C940 00000000 00000000  0500035C C1D7C940 00000000 00000000
*...*API ...........*API ........*
                  02A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  02C0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*................................*
                  02E0  00000000 00000000 00000000 00000000  4C4CE399 81838540 C5958440 40404C4C
*................<<Trace End   <<*

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

              TASK-00035 KE_NUM-003F TCB-L8000/007C6B90 RET-99902F92 TIME-08:32:00.7697405327
INTERVAL-00.0000056718        =000172=
                1-0000  01
*.                      *
                2-0000  C4E2D5C3 E2D8D340
*DSNCSQL                *
                3-0000  B60AF030 A54A8EC0                                                *..0.v$.
{                        *
                4-0000  18CCB654 0006F004 19044204 00000000  007A8000 007A8000 18F0005C 18F0CCD0
*......0.........:...:...0.*.0.}*
                  0020  190441C0 19044210 19044206 18F000D0  19044200 18CCBCA0 190441C8 190441B0  *...
{.........0.}...........H....*
                  0040  190441AC 190441EC 18CCB5F3 190441FD  18CCB5F4 190441B4 190441B2 00020000
*...........3.......4...........*
                5-0000  D3F8
*L8                     *

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

              TASK-00035 KE_NUM-003F TCB-L8000/007C6B90 RET-99902F92 TIME-08:32:00.7697463608
```

```
INTERVAL-00.0000058281        =000173=
                1-0000  01
*.                             *
                2-0000  C4E2D5C3 E2D8D340
*DSNCSQL                       *
                3-0000  B60AF030 A54A8EC0                                                  *..0.v$.
{                               *
                4-0000  E3C5E2E3 D7F0F540 00000001 00100000  19900000 999000E0 000114E0 00000000
*TESTP05 ............r..\...\....*
                  0020  00000020 00007BB0 00000000 98400000  0040000A 00101100 00000000 00000000
*......#.....q ... ..............*
                  0040  00000000 00000000 00000000
*............                   *
```

CSUB trace

The CSUB is the connection control block that CICS uses to manage, and exchange information about, a thread into Db2. It is the CICS equivalent to the Db2 connection control block. The CSUB is linked to the thread TCB that CICS uses to run the thread.

See Overview: How CICS connects to Db2 for a full explanation of the relationship between the CSUB, the Db2 connection control block and the thread TCB.

In addition to using CICS tracing, DFHD2D2 maintains a 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 15 on page 144.

Table 15. Layout of CSUB trace table entry

Bytes	Content	Information
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 ASSO - Associate request COMM - Commit request CTHD - Create thread request DISS - Dissociate request ERRH - Error handler request IDEN - Identify request PREP - Prepare request PSGN - Partial sign-on request SIGN - Full sign-on request SYNC - Single phase request TERM - Terminate thread request TIDN - Terminate identify request TSGN - Terminate sign-on request *REC - Recovery routine entered
Bytes 8-9	reserved	
Bytes 10-11	frb or ifc return code	Two-byte return code
Bytes 12-15	frb or ifc reason code	Four-byte reason code

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

Figure 38 on page 145 shows that an identify thread, a signon thread, and a create thread have been issued to Db2. There is an API request, followed by a syncpoint and a dissociate (which dissociates the Db2 connection control block from the L8 TCB). The transaction now makes another API request, starting another unit of work, and the Db2 connection control block is reassociated (ASSO) with the L8 TCB. A partial sign-on occurs to create an accounting record for the previous unit of work. The API request is now issued to Db2.

```
6-0000  03206EC4 C6C8C4F2 C3E2C240 40404040  CFE6D380 716D6B32 2B0A1300 2B0D36C0
*..>DFHD2CSB     .WL.._,........{*
  0020  2C6A8370 0098DC88 00000000 00000000  CFE6D381 5F3C62FE 00000000 00000000
*..c..q.h.........WLa¬...........*
  0040  2B0D372C 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*.............................*
  0060  00000000 00000000 00000000 E3C5E2E3  D7F0F540 C3C9C3E2 E4E2C5D9 40404040  *............TESTP05
CICSUSER    *
  0080  40404040 C5D5E3D9 E7D7F0F5 F0F0F0F1  00000000 CFE6D380 742927E2 C7C2C9C2  *
ENTRXP050001.....WL....SGBIB*
  00A0  D4C9E8C1 C9E8C3E6 E3F1F5F0 E6D37E22  16AC8000 44A00000 00000001 C3C9C3E2
*MIYAIYCWT150WL=.............CICS*
  00C0  E4E2C5D9 40404040 40404040 40404040  40404040 40404040 40404040 FF4EA000
*USER                        .+..*
  00E0  C6D9C240 00030001 2AF0DCB0 00000000  00000000 00000000 00180A05 00010000
*FRB .....0.....................*
  0100  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*.............................*
  0120  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*.............................*
  0140  00000000 00000000 00000000 00000000  00000000 00000000 ACB8C110 00000000
*.......................A.....*
  0160  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*.............................*
  0180  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*.............................*
  01A0  00000000 00000000 00000000 00000000  40404040 40404040 40404040 40404040
*.............               *
  01C0  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040
*                            *
  01E0  40404040 00000001 00000048 42600048  2C6511B8 00000000 00000000 00000000
*    ........-.................*
  0200  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*.............................*
  0220  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
*.............................*
  0240  00000000 00000000 00000000 00000000  00000000 00000000 0000000C 2CB8C2B0
*...........................B.*
  0260  6E6EE399 81838540 E2A38199 A3406E6E  0100067C C9C4C5D5 00000000 00000000  *>>Trace Start
>>....IDEN........*
  0280  0200067C E2C9C7D5 00000000 00000000  0300067C C3E3C8C4 00000000 00000000
*....SIGN............CTHD........*
  02A0  0400067C C1D7C940 00000000 00000000  0500067C E2E8D5C3 00000000 00000000
*....API ............SYNC........*
  02C0  0600067C C4C9E2E2 00000000 00000000  0700067C C1E2E2D6 00000000 00000000
*....DISS............ASSO........*
  02E0  0800067C D7E2C7D5 00000000 00000000  0900067C C1D7C940 00000000 00000000
*....PSGN............API ........*
  0300  00000000 00000000 00000000 00000000  4C4CE399 81838540 C5958440 40404C4C  *..............<
```

Figure 38. Sample CSUB trace — CICS connected to Db2

CSUB Abend information

If a CICS-Db2 subtask abends, as part of the recovery process, CICS saves information from the MVS SDWA in the CSUB control block.

For example, the CSB_SDWA_REGS (regs 0 -15) and the CSB_SDWA_PSW fields. The CSB_SDWA_REGS is the 16 words following the SDWA eye catcher. The CSB_SDWA_PSW field is the two words following the CSB_SDWA_REGS field. Although the CSUB is typically cleared by the FREEMAIN function following the failure, the exception trace written at the time of failure, AP 319D, captures the CSUB control block containing the SDWA information.

Dump for CICS Db2

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 can 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 39 on page 146. The sample provides information on the global state of the CICS Db2 connection, and a summary of transactions (under the headings Tran id, Task num, TcaAddr, TieAddr, LotAddr, Rctename, RcteAddr, CsubAddr, Correlation id, Uowid, Subtask running, and TCB in DB2). This sample was output when CICS was connected to DB2 Version 6 using the open transaction environment.

```
===DB2: CICS/DB2 - SUMMARY
==DB2: GLOBAL STATE SUMMARY
   Db2conn name:                          RCTJT
   Connection status:                     Connected
   In standby mode:                       No
   DB2 id:                                DE2D
   DB2 Group id:
   DB2 release:                           0610
   Operating in OpenAPI mode:             Yes
   Service task started:                  Yes
   Main subtask started:                 No - not required
   Tcb limit:                             12
   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  TCB
id   num                                          id                          running  in DB2
-------------------------------------------------------------------------------------------------------
XP05 00050 184AB680 190442F0 19044370 XP05     18DF2D78 18D10330 ENTRXP050002 B60A17A16316E1C7 N/A      Yes
XP05 00048 184AC680 19044190 19044210 XP05     18DF2D78 18D10030 ENTRXP050001 B60A1794A7A9E7C4 N/A      No
```

Figure 39. Dump example

Db2 thread identification

A thread executing in Db2 on behalf of a CICS transaction is identified by its correlation ID, which is set by the CICS Db2 Attachment Facility.

Db2 allows up to 12 bytes to be used for the correlation ID.

The format 12 byte correlation ID 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 sign-on to Db2. If sign-on reuse occurs by a thread using a primary authorization ID which remains constant across multiple transactions (for example, by using AUTHID(name)) only one sign-on 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 sign-on occurred.

Transaction abend codes for CICS Db2

The CICS Db2 attachment facility uses multiple abend codes, each of which is unique to a particular error.

The transaction abend codes are AD2x or AD3x. They are documented in Transaction abend codes and are available using the CICS-supplied transaction, CMAC.

Execution Diagnostic Facility (EDF) for CICS Db2

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 Db2 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 40 on page 147, and Figure 41 on page 148.

```
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 40. 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 41. EDF example of the "after" SQL EXEC panel

Handling deadlocks in the CICS Db2 environment

Deadlocks can occur in a CICS Db2 system between two or more transactions or between one transaction and another Db2 user. Deadlocks can involve one or two resources.

About this task

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. Give special attention to deadlock in the following situations:

- Other transactions are often delayed because they access resources held by the partners in the deadlock. This increases the response times for these transactions, which can result in a cascade.
- 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. When DROLLBACK(YES) is specified, the attachment facility initiates a SYNCPOINT ROLLBACK before returning control to the application. Additionally the attachment facility changes the SQL return code returned by Db2 from -913 to -911, and returns -911 to the application. 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.

Related concepts

"Troubleshooting Db2" on page 135
This section discusses problem determination for CICS Db2.

"Thread TCBs (task control blocks)" on page 135
In the CICS Db2 environment, each thread into Db2 runs under a thread task control block (TCB). This section provides further technical information about thread TCBs, to assist with problem determination.

"Wait types for CICS Db2" on page 135
The CICS Db2 task-related user exit, DFHD2EX1, issues **WAIT_MVS** calls for different purposes. These calls are distinguished by their resource name and type values. The resource name and resource type values are visible in the dispatcher section of a CICS system dump and also appear on the **CEMT INQUIRE TASK** panel as Hty and Hva values respectively.

"Messages for CICS Db2" on page 137
Messages issued by the CICS Db2 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.

"Trace for CICS Db2" on page 138
The CICS Db2 attachment facility uses AP domain trace points in the range 3100 to 33FF.

"Dump for CICS Db2" on page 146
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.

"Db2 thread identification" on page 146
A thread executing in Db2 on behalf of a CICS transaction is identified by its correlation ID, which is set by the CICS Db2 Attachment Facility.

"Transaction abend codes for CICS Db2" on page 146
The CICS Db2 attachment facility uses multiple abend codes, each of which is unique to a particular error.

"Execution Diagnostic Facility (EDF) for CICS Db2" on page 147
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.

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. The main purpose of update (U) locks is 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.

Detecting deadlocks

Deadlock information is available in the Db2 performance trace or in the MVS log.

About this task

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 42 on page 150.

```
DSNT375I PLAN p1 WITH CORRELATION ID id1
AND CONNECTION ID id2 IS DEADLOCKED with
PLAN p2 WITH CORRELATION ID id3
AND CONNECTION ID id4.

DSNT501I DSNILMCL RESOURCE UNAVAILABLE
         CORRELATION-ID=id1,CONNECTION-ID=id2
         REASON=r-code
         TYPE name
         NAME name
```

Figure 42. 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 might 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.

Limit the Db2 performance trace to the two plans indicated in the MVS log message. Limiting the trace to the two CICS transaction IDs (authorization IDs) involved can also be reasonable. Include `class(06)` for general locking events and `class(03)` for SQL events in the performance trace . The Db2 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.

About this task

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 the SQL statements and the resources involved in the deadlock, and finding the access path will typically show you why the deadlock occurred. Knowledge of why the deadlock occurred can help you to develop a solution. However; this process can be time-consuming.

Resolving deadlocks

You must try to understand why deadlock occurred so that you can choose the right action to solve the deadlock.

About this task

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.

Table 16 on page 151 shows a list of preventive actions and the corresponding main effects.

Table 16. Deadlock prevention

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

Notes:

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
- 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.

Notices

This information was developed for products and services offered in the United States of America. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 rights 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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the 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.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want 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 Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119 Armonk,
NY 10504-1785
United States of America

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 Client Relationship Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming interface information

IBM CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 6 (CICS TS 5.6) are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- CICS TS security
- Developing for external interfaces
- Application development reference
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS TS 5.6, but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- CICS TS diagnostics reference

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS TS 5.6 are included in the following manuals:

- Application Programming Guide and Application Programming Reference
- Business Transaction Services

- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- CICS Transactions
- CICSPlex System Manager (CICSPlex SM) Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS TS 5.6, but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Apache, Apache Axis2, Apache Maven, Apache Ivy, the Apache Software Foundation (ASF) logo, and the ASF feather logo are trademarks of Apache Software Foundation.

Gradle and the Gradlephant logo are registered trademark of Gradle, Inc. and its subsidiaries in the United States and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Red Hat®, and Hibernate® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Spring Boot is a trademark of Pivotal Software, Inc. in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Zowe™, the Zowe logo and the Open Mainframe Project™ are trademarks of The Linux Foundation.

The Stack Exchange name and logos are trademarks of Stack Exchange Inc.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability
These terms and conditions are in addition to any terms of use for the IBM website.

Personal use
You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights
Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, (*Software Offerings*) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information (PII) is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect PII. If this Software Offering uses cookies to collect PII, specific information about this offering's use of cookies is set forth below:

For the CICSPlex SM Web User Interface (main interface):
Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name and other PII for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other PII for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):
Depending upon the configurations deployed, this Software Offering may use session cookies that do not collect PII. These cookies cannot be disabled.

For CICS Explorer:

Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user’s user name and password, for purposes of session management, authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect PII from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled *Cookies, Web Beacons and Other Technologies* and the IBM Software Products and Software-as-a-Service Privacy Statement.

Index

A

B

C

D

E

F

G

H

T

U

V

W

X

Z