CICS Transaction Server for z/OS
Version 4 Release 2

IMS Database Control Guide

SC34-7170-01

CICS Transaction Server for z/OS
Version 4 Release 2

IMS Database Control Guide

SC34-7170-01

Note
Before using this information and the product it supports, read the information in "Notices" on page 159.

This edition applies to Version 4 Release 2 of CICS Transaction Server for z/OS (product number 5655-S97) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Preface

Who this manual is for

This manual is for anyone who uses the CICS®-IMS™ Database Control interface, referred to as DBCTL in the rest of this manual.

This manual documents intended Programming Interfaces that allow the customer to write programs to obtain the services of Version 4 Release 2.

This manual is intended to help you understand DBCTL. It contains guidance on evaluating, installing, and using DBCTL. This manual also discusses migration from local DL/I.

For programming information on programming interfaces provided by IMS, see *IMS Application Programming: EXEC DLI Commands* and *IMS Application Programming: DL/I Calls*.

What this manual is about

The aim of this manual is to give introductory and guidance information on evaluating, installing, and using DBCTL.

This manual is intended to be used in conjunction with existing manuals in the CICS and IMS libraries, to which it refers where appropriate.

What you need to know before reading this manual

Before you read this manual, you need a general understanding of CICS and IMS. You should also have some knowledge of the concepts of data management and databases.

How to use this manual

Aspects of DBCTL, from installation through performance considerations, are presented in the order in which you are likely to need them.

Terms used

In general, this manual refers to Customer Information Control System and Information Management System as “CICS” and “IMS”, respectively. CICS used without qualification normally refers to the CICS element of CICS Transaction Server for z/OS®.

Changes in CICS Transaction Server for z/OS, Version 4 Release 2

For information about changes that have been made in this release, please refer to *What's New* in the information center, or the following publications:

- *CICS Transaction Server for z/OS What's New*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 4.1*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.2*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.1*

Any technical changes that are made to the text after release are indicated by a vertical bar (|) to the left of each new or changed line of information.

Chapter 1. Overview of Database Control (DBCTL)

CICS can access DL/I databases with the CICS-DBCTL interface or by using remote DL/I.

Using DBCTL
: This is when DBCTL satisfies the DL/I request issued from the CICS system with the CICS-DBCTL interface.

  Installing and using DBCTL are introduced in this manual (but note that you also must refer to other CICS and IMS manuals for further information).

Using remote DL/I
: Remote DL/I is done with CICS function shipping a DL/I request to another CICS system, in which the DL/I support can be remote DL/I or DBCTL. See the *CICS Intercommunication Guide* for more information about function shipping, and the *CICS Transaction Server for z/OS Installation Guide* for information about adding remote DL/I support.

Note:

1. Although these methods of accessing DL/I databases can coexist, a program specification block (PSB) can only contain databases that are controlled by one of the methods.
2. CICS Transaction Server does not support local DL/I.

CICS can also access DL/I databases in an IMS Database Manager/Transaction Manager (IMS DM/TM) system using the CICS-DBCTL interface. This means that you can have access to DL/I databases controlled by IMS DM/TM without needing to use IMS data sharing, if CICS and IMS DM/TM are in the same MVS™ image.

Figure 1 on page 2 illustrates the three kinds of DL/I request.

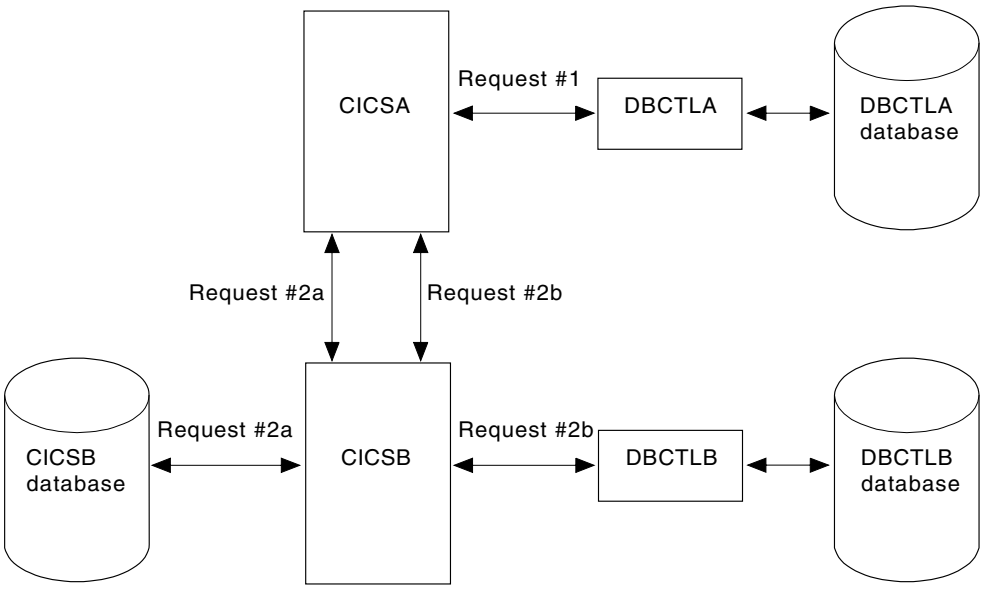


Figure 1. DL/I request handling within CICS

Note:

1. Request #1 is a DBCTL request from CICS A to DBCTL A for a database controlled by DBCTL A. See "CICS-DL/I router (DFHDLI)" on page 3 for a description of request processing.
2. Requests #2a and #2b are two separate remote (function shipped) DL/I requests to databases controlled by, or connected to, other CICS systems (which can be in the same MVS image or a different one). There are two ways of issuing such requests:
   - Request #2a from CICS A to CICS B for a database controlled by CICS B.
   - Request #2b from CICS A to CICS B for a database controlled by DBCTL B. The most likely reason for using request #2b is if CICS A and CICS B are in different MVS images.

Connecting to DBCTL

You can connect to, and disconnect from, DBCTL using the CICS-supplied transaction CDBC.

When you have connected to DBCTL by means of CDBC, you can issue DL/I requests from your application programs. There is another CICS-supplied transaction, CDBI, which you can use to inquire on the status of the connection to DBCTL from CICS. See "Connection, disconnection, and inquiry transactions for the CICS DBCTL interface" on page 37 for information on using CDBC and CDBI.

CICS-IMS DBCTL environment

This figure summarizes the components of a CICS-DBCTL interface.

Figure 2 on page 3 gives an overview of a CICS-DBCTL interface. Each box represents an address space running within a single MVS system. The marked area between the second CICS and the first BMP is the point at which CICS components end and IMS components begin.

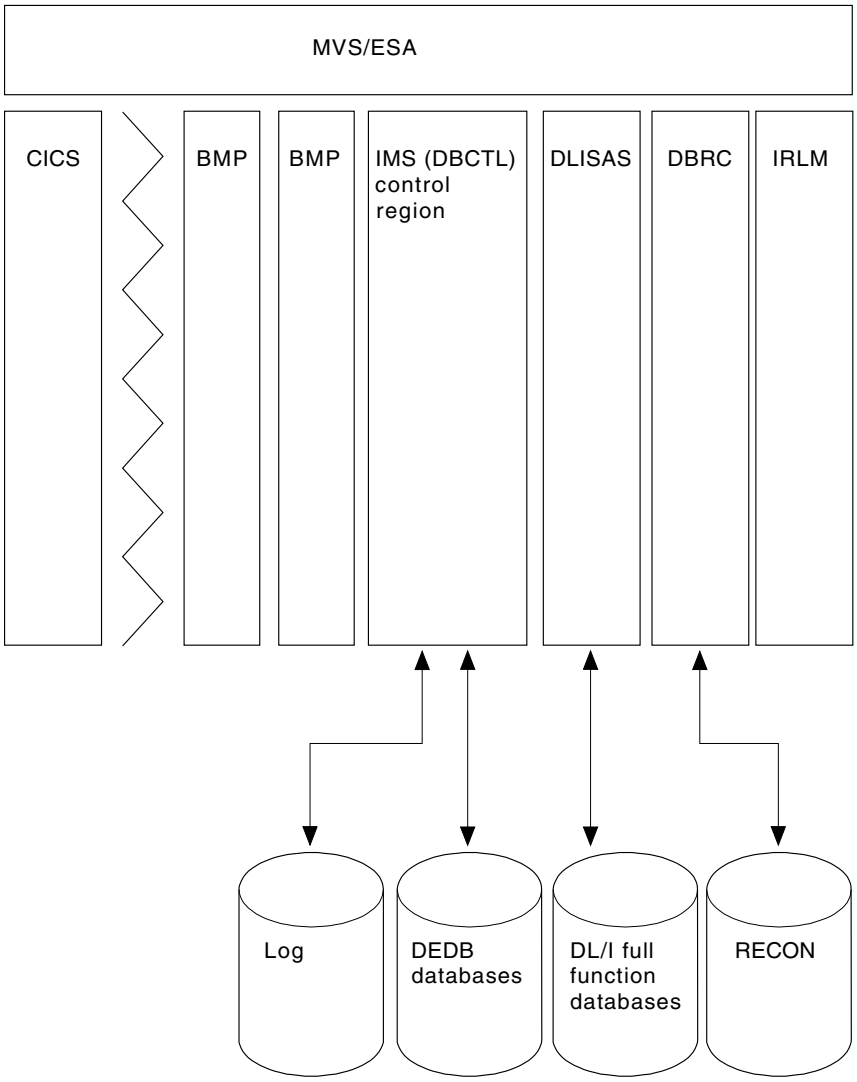


Figure 2. CICS-DBCTL interface

CICS-IMS DBCTL environment: description of components

The following topics give detailed information about each of the major components of the CICS-IMS DBCTL interface. See "Summary of DBCTL components in CICS and IMS" on page 5 for an illustration of these components.

CICS-DBCTL interface control components in CICS address space

The components of the CICS-DBCTL interface in the CICS address space are: the CICS-DL/I router (DFHDLI), the CICS database adapter transformer (DFHDBAT), and the database resource adapter (DRA).

CICS-DL/I router (DFHDLI)

The CICS-DL/I router, DFHDLI, forms the interface between your application programs and the DL/I call processor. DFHDLI accepts requests for remote or DBCTL database processing. If the request is for DBCTL, DFHDLI passes the request to the CICS-DL/I DBCTL processor, DFHDLIDP. The request then goes to

the task-related user exit interface and the CICS database adapter transformer, DFHDBAT. The task-related user exit interface is also referred to as the resource manager interface (RMI). For more information about the task-related user exit interface, see the *CICS Customization Guide*.

CICS database adapter transformer (DFHDBAT)

The main responsibility of the CICS database adapter transformer, DFHDBAT (also referred to in IMS publications as the adapter, or adapter/transformer) is to communicate with the database resource adapter (DRA). DFHDBAT constructs parameter lists for the DRA. These parameter lists enable CICS to connect to and disconnect from DBCTL, and enable DL/I requests to be processed. To summarize, DFHDBAT performs the following tasks:

- Tells the DRA that it must initialize the interface to DBCTL in response to a request from the connection program (DFHDBCON).
- Tells the DRA when it must issue PSB schedule requests, DL/I requests, and sync point requests in response to a request from the CICS-DBCTL processor (DFHDLIDP).
- Tells the DRA that it must terminate the interface to DBCTL in response to a request from the disconnection program (DFHDBDSC). If an orderly disconnection has been requested, DFHDBAT ensures that all current CICS tasks that use DBCTL complete before telling the DRA to terminate the interface. If an immediate disconnection has been requested, DFHDBAT ensures that only the current CICS-DL/I requests that use DBCTL can complete before telling the DRA to terminate the interface.

CICS master terminal operators can use the CICS-supplied transaction CDBC to connect to and disconnect from DBCTL. They can also automate connection to DBCTL, as described in "Connecting to DBCTL: overview" on page 35.

DFHDBAT is defined as a threadsafe program.

Database resource adapter (DRA)

The database resource adapter (DRA) performs the following tasks:

- Requests connection to, and disconnection from, DBCTL.
- Tells CICS when a shutdown of DBCTL has been requested, or if DBCTL has failed.
- Manages threads. A CICS application thread provides a two-way link between an application and DBCTL. When a CICS transaction issues a DL/I request to DBCTL, the thread represents that CICS transaction in DBCTL. It identifies the existence of the transaction, traces its progress, sets aside the resources it needs to be processed, and delimits its accessibility to other resources.
- Establishes contact with the DBCTL address space and loads the DRA startup parameter table. The DRA startup parameter table provides the parameters needed to define the interface to a DBCTL subsystem. (See "Defining the IMS DRA startup parameter table" on page 27, for a list of DRA startup table parameters.)

Components of DBCTL in IMS address spaces

The components of DBCTL that reside in IMS address spaces are: the DBCTL subsystem, the DL/I separate address space (DLISAS), the Database Recovery Control (DBRC) facility, and the internal resource lock manager (IRLM).

DBCTL

The DBCTL subsystem contains support and features required to process full function DL/I databases and DEDBs. Full function supports HSAM, SHSAM, HISAM, SHISAM, HDAM, and HIDAM databases. Each DBCTL subsystem is made up of three address spaces: DBCTL, DLISAS, and DBRC. A single DBCTL can service multiple CICS systems, but a CICS system can connect to only one DBCTL at a time. A CICS system can connect to one DBCTL, disconnect from it, and then connect to a different DBCTL.

DL/I separate address space (DLISAS)

DL/I separate address space (DLISAS), which is **required** with DBCTL, is a separate address space that contains DL/I code, control blocks, buffers for DL/I databases.

Database Recovery Control (DBRC)

Database Recovery Control (DBRC) is an IMS facility that supports log management, recovery control, and database sharing by providing the necessary information to subsystems, batch programs, and utilities. DBRC is required with DBCTL for log control and can optionally be used for database recovery control and data sharing. See "Database recovery control (DBRC)" on page 69 for information about DBRC and logging, and *IMS Operations Guide* for more general information about using DBRC.

Internal resource lock manager (IRLM)

The internal resource lock manager (IRLM) is a global lock manager that is a feature of IMS and resides in its own address space.

IRLM is the preferred lock manager for DBCTL. For more information about locking using IRLM see *IMS System Administration Guide* or *IMS Administration Guide: System*.

Summary of DBCTL components in CICS and IMS

These are the major components in a simple CICS-IMS DBCTL environment. Each separate box represents an address space. All the components shown except the IRLM are mandatory.

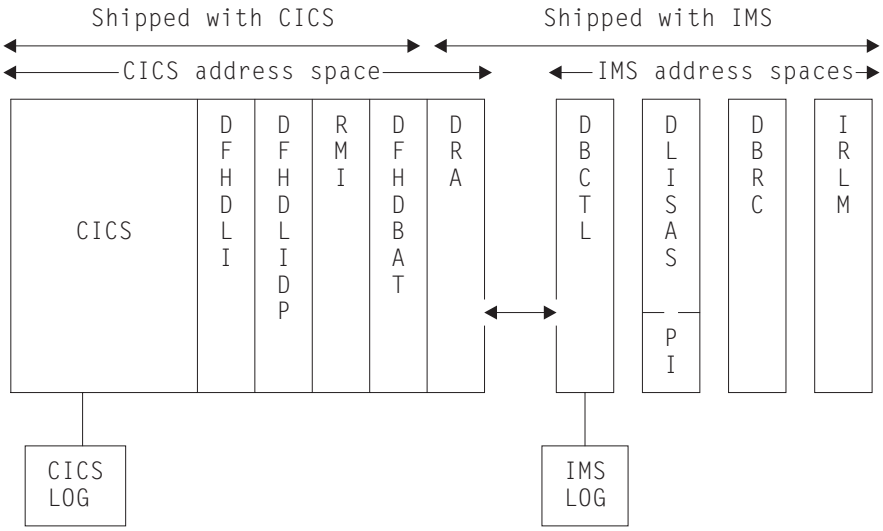


Figure 3. Major components of a simple CICS-IMS DBCTL environment

Coordinator control subsystem (CCTL)

The coordinator control subsystem (CCTL) is the transaction management subsystem that communicates with the DRA, which in turn communicates with DBCTL.

In a CICS-DBCTL environment, the CCTL is CICS. The term CCTL is used in a number of DBCTL operator commands and in the IMS manuals. CICS users of DBCTL should take the term CCTL to mean a CICS system that is attached to IMS by means of DBCTL.

Resources you can access from a CICS environment that includes DBCTL

This diagram shows you the resources you can access from a CICS environment that includes DBCTL.

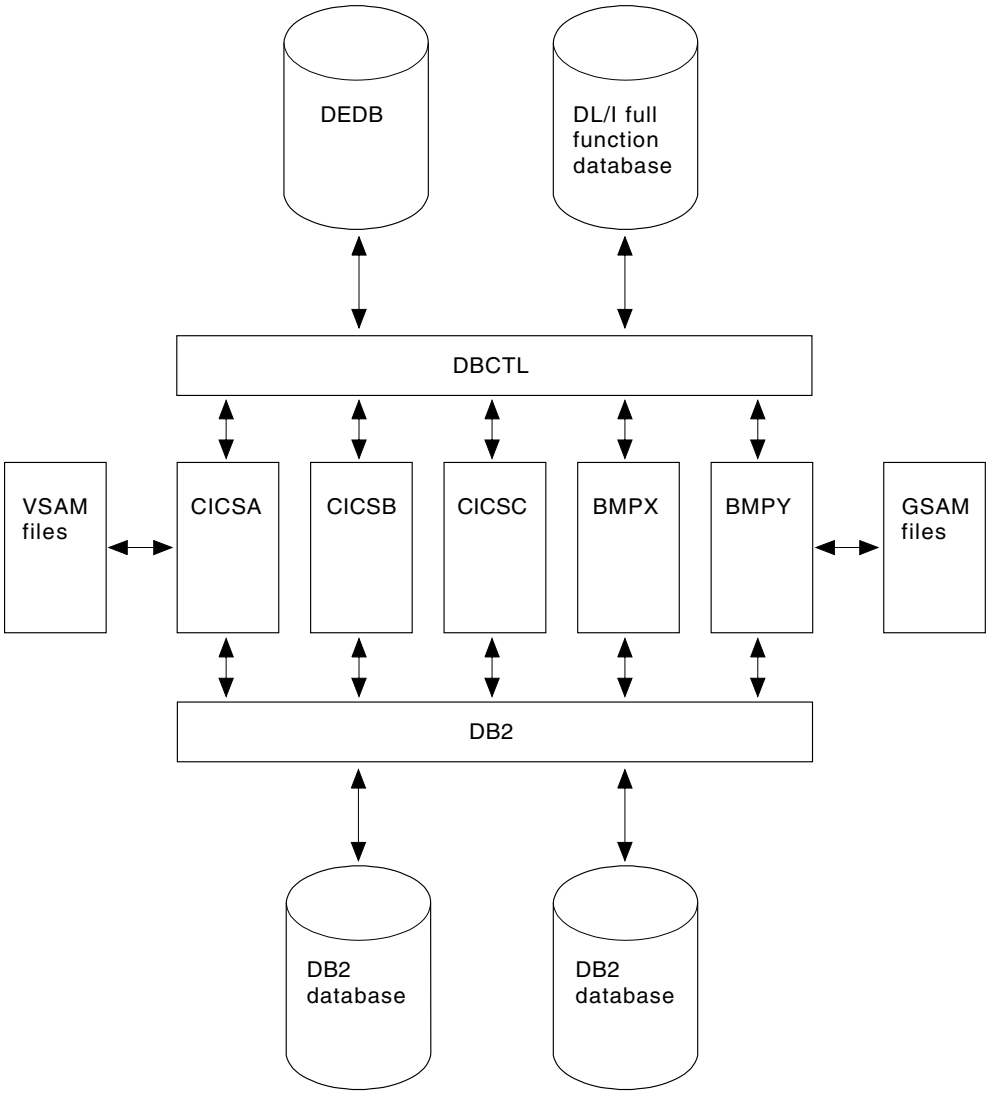


Figure 4. Resources you can access from a CICS environment that includes DBCTL

A single CICS task can use DB2® tables, IMS databases (using DBCTL or remote DL/I), and CICS-managed local or remote resources (for example, VSAM files).

The CICS-DB2 and the CICS-DBCTL interfaces are similar in that they both use the task-related user exit interface, and have a two-phase commit process. However, they differ in a number of respects. For example, CICS supports DBCTL and remote DL/I, and must determine at PSB schedule time which of them is being used.

System service requests

Your CICS application programs can use these IMS system service requests in addition to those related to data availability.

- DEQ (in its command or call format) releases segments that were retrieved using the LOCKCLASS keyword or the Q command code. LOCKCLASS and Q enable an application program to reserve segments for its use.

- LOG (in its command or call format) can be used to write a record from an application program to the IMS log. You may prefer to use this instead of EXEC CICS journal commands so that all your DBCTL information is on the IMS log instead of the CICS log.

See Chapter 5, "Application programming for DBCTL," on page 85 for more information on using these requests.

Access to data entry databases (DEDBs)

Data entry databases (DEDBs) provide the same features as HDAM databases (with the exceptions of secondary indexing and logical relationships).

They also have a number of advantages. Using DEDBs enables you to have very large databases with high availability. DEDBs are designed to provide efficient storage and fast online gathering, retrieval, and update of data, using VSAM entry sequenced data sets (ESDSs).

DEDBs are hierarchic databases that can contain up to 127 segment types. One of these segments is always a root segment. The remaining 126 segments can either be direct dependent (DDEP) segments, or 125 DDEP segments and one sequential dependent (SDEP) segment. A DEDB structure can have as many as 15 hierarchical levels.

DEDBs are made up of database records stored in a set of up to 240 areas. Each area contains a range of database records (which you can specify using the DEDB randomizing routine) that contain the entire logical structure for a set of root segments and their dependent segments. Areas are independent of each other, are individually recognized, can be accessed by multiple programs and DEDB utilities, are the basis for recovery procedures, and are largely transparent to application programs.

DEDBs provide the following advantages:

- Large databases
  - Areas can be as large as 4 gigabytes, and because you can have up to 240 areas in a single database, you can use very large databases, which you would have to partition if you were not using DEDBs.
- Flexible design
  - Each area can be designed to meet your storage, availability, performance, and application needs. Areas can be separately reorganized and reacquired.
  - You use the DEDB direct reorganization utility to physically reorganize DEDBs to reduce ESDS fragmentation without taking them offline.
- Increased data availability
  - If a DEDB area is not available, a PSB requiring that database can still be scheduled provided the area it requires is not the one that is unavailable and, of course, the database itself is available. A PSB that requires an unavailable area is still scheduled, and receives a status code indicating the condition. You can therefore delay recovery until it is convenient to take the area offline.
  - You can have up to seven copies of the same area. Each copy is called an area data set (ADS) and all are automatically maintained in synchronization. This is called multiple area data set (MADS) support. Write operations are done to each ADS, but read operations are done from only one ADS. With MADS, read and write errors are much less common because, if data cannot be read from, or written to, the first copy, the next copy will automatically be used.

Read errors are transparent to application programs (except in the rare instance where a read operation is unsuccessful with all ADSs).

- You can use DEDB utilities, which are run on an area basis and can be run online concurrently with online update. This helps to reduce the time for which areas have to be taken offline. For example, you can avoid using offline database recovery by using the DEDB area data set create utility. This online utility makes a new corrected copy of an area from existing copies of that area. It creates one or more copies from multiple DEDB ADSs during online transaction processing, enabling application programs to continue while the utility is running.
- You use the DEDB initialization utility to initialize one or more data sets or one or more areas of a DEDB offline.
- You can use the DEDB area data set compare utility if you suspect you may have problems with compatibility of data. It compares control intervals (CIs) of different copies of an area, and lists all the CIs that do not have equal content. In the case of unequal comparison, full dumps of up to ten unmatched CIs are printed out on the device you have specified.

- Efficient data retrieval and entry
  - DEDB attempts to physically write DDEP segments hierarchically in the same CI as the parent segment, which can make retrieval faster.
  - The SDEP segment (located at the end of the ADS) is designed especially for fast, online, mass insert in applications such as data collection, auditing, and journaling. This is because SDEP segments for an area are stored rapidly, regardless of the root on which they are dependent. For example, in a banking application, transaction data can be collected during the day and inserted as SDEPs in an account database. At the end of the day, these transactions can be reprocessed by first retrieving them using the sequential dependent scan utility. This online utility retrieves SDEP segments in mass and copies them to a sequential data set. You can then process this data set offline using your own programs; for example, for a statistical analysis. The area involved remains available while the utility is running.
  - You can delete SDEPs using the DEDB sequential dependent delete utility, which deletes SDEP segments within a specified limit of a DEDB area.
  - The ability to use high speed sequential processing (HSSP). HSSP is useful with applications that do large scale sequential updates to DEDBs. HSSP can reduce DEDB processing time, enables an image copy to be taken during a sequential update job, and minimizes the amount of log data written to the IMS log. For further guidance, see “High speed sequential processing (HSSP)” on page 157.
- Improved performance
  - Pathlength is reduced because DEDBs use the MVS Data Facility Product (MVS/DFP) Media Manager offering.
  - You can improve speed of access, or concurrent access, to DEDBs by tuning DEDB buffer pool specifications. (See “DEDB performance and tuning considerations” on page 155.)
  - Logging overhead is reduced because only after-images are logged and because logging is done during syncpoint processing only.
  - The amount of I/O needed for each SDEP segment inserted can be very low, because SDEPs are gathered from various transactions, stored in last-in first-out order in one buffer, and are written out only when that buffer is full. This means that many transactions “share the cost” of SDEP writes.

- Most DEDB processing is done in parallel to allow multithreading. Writes to the database are done by a number you specify (up to 255) of parallel processes called output threads. Furthermore, the DEDBs are not updated during application program processing, but the updates are kept in buffers until a syncpoint occurs. (See "When updates are written to databases" on page 70.) This means that waiting applications can be processed sooner and improves throughput on multiprocessors.
- DEDBs have their own resource manager and normally need to interact very infrequently with program isolation or the IRLM (unless you are using block level sharing). DEDBs maintain their own buffer pool.
- You can use subset pointers in your application programs to speed up processing. A major problem in some applications is the need to process long twin chains of segments. Occasionally database design must be modified because some database records have excessively long twin chains. Subset pointers give direct access to subsets of long twin chains of segments, which can speed up application processing because segments located in front of the subset do not have to be searched. Each pointer points to the first occurrence of a subset in a range of direct dependent segments. See "Command codes to manage subset pointers in DEDBs" on page 90 and "Keywords and corresponding command codes" on page 92 for information about using subset pointers in application programs. (See *IMS Database Administration Guide* or *IMS Administration Guide: Database Manager* for guidance on database structure.)

Online image copy utility

The online image copy utility is used to create an as-is copy of your database while it is being updated. The copy can then be used for recovery purposes. This utility is used for HISAM, HDAM, and HIDAM databases only.

Online change utility

In many installations, it is important for the online system to be is available to users for most of the day.

The online change utility enables you to update ACBLIBs, which contain PSBs and data management blocks (DMBs), and security information belonging to full function databases, without bringing down the system. For guidance information on this utility, see *IMS System Administration Guide* and *IMS Utilities Reference: Database*.

Online reorganization for DEDBs

The data entry database (DEDB) direct reorganization utility enables you to reorganize DEDBs without taking them offline.

For more information see "Access to data entry databases (DEDBs)" on page 8.

Chapter 2. Installing DBCTL, and defining CICS and IMS system resources

This section describes how to install DBCTL and define CICS and IMS system resources.

The *IMS Release Planning Guide* has information on release compatibility for CICS and IMS.

Installing and generating DBCTL

Install DBCTL, verify the installation, and connect CICS to DBCTL.

Before you begin

Before installing and generating DBCTL you must have CICS Transaction Server for z/OS, Version 4 Release 2 and IMS installed. Check the program directory for any PTFs or APARs that need installing. Develop your own procedures for installing DBCTL, depending on the DBCTL facilities you want to use. For more information about IMS installation and system definition, see the*IMS Installation Guide*, the *IMS System Definition Reference*, and the *IMS Installation Volume 2: System Definition and Tailoring*.

About this task

Follow these steps to install DBCTL, verify the installation, and connect CICS to DBCTL.

Procedure

1. Prepare a PDIR that does not specify PSBs. For more information, see "PSB directories (PDIRs)" on page 14.
2. Update system procedure libraries; for example, SYS1.PROCLIB, with the startup procedures for DBCTL, DLISAS, DBRC, and the IRLM if you are using it. These startup procedures can be found in the IMS.PROCLIB library.
3. Use the CICS supplied DBCTL-installation verification procedure, DFHIVPDB, to check that: DBCTL has been fully installed, CICS has integrated with MVS, and that all required online data sets have been allocated and initialized. For more information, see the *CICS Transaction Server for z/OS Installation Guide*.
4. You must use ACB generation to create members of the IMS.ACBLIB. Failure to carry out this step can cause user errors.
5. If you plan to use dynamic allocation, create DFSMDA members. For more information, see "IMS dynamic allocation macro (DFSMDA)" on page 25.
6. Start DBCTL; DBCTL then issues a start command for DLISAS and DBRC.
7. Verify that DBCTL recognizes the PSBs and DBDs you defined in the DBCTL generation, you can check this using the DBCTL operator command /DISPLAY. For more information, see "Finding out current status of DBCTL activities" on page 55.

8. Check that your log archiving setup works before doing any more testing. If your log archiving is not set up it is possible for the logs to fill and stall your system. For more information about setting your log archive setting, see “Log control with DBRC” on page 23.
9. Assemble a database resource adaptor (DRA) to connect CICS to DBCTL. For more information, see “Defining the IMS DRA startup parameter table” on page 27.
10. Start CICS and test the connection to DBCTL, using the CDBC transaction. For more information, see “CDBC transaction for connect and disconnect” on page 37.
11. Generate an initialization PLT, so that CICS can connect to DBCTL automatically at startup time. For more information, see “Connecting DBCTL to CICS automatically” on page 36.
12. Test the applications that you defined to DBCTL.
13. Set up and test recovery and restart of CICS and DBCTL, and database recovery. For more information, see Chapter 4, “Recovery and restart operations for DBCTL,” on page 65.

Defining CICS system resources for DBCTL

Use this information to help you define system resources for DBCTL.

System initialization parameters

The CICS system initialization parameters contain information needed to initialize and control system functions and the initialization process.

It also contains module suffixes to enable you to choose between different versions of CICS modules and tables. You can generate several SITs and select the one that best meets your current requirements at initialization time. If you have more than one CICS system, each can use a different SIT.

Specifying DL/I support in system initialization parameters

In CICS Transaction Server for z/OS, Version 4 Release 2, there is no DLI system initialization parameter. Support for DBCTL is always present. Support for remote DL/I is included if the PDIR system initialization parameter is specified.

Note: The default is PDIR=NO, meaning that by default support for remote DL/I is not included.

Related information:

PDIR system initialization parameter in the System Definition Guide

Reviewing CICS system initialization parameters

With DBCTL, many CICS system initialization parameters are replaced by DBCTL generation parameters. You must change what you specify for others, because DL/I code has been removed from the CICS address space.

Table 1 on page 13 lists the CICS system initialization parameters relevant to DL/I. It states whether each parameter applies to DBCTL or remote DL/I (in the **D** and **R** columns, respectively). Where applicable, it lists the corresponding IMS startup parameter that applies to DBCTL. Finally, it mentions special considerations for DBCTL.

See “Generating DBCTL” on page 17 for more information about the IMS and DBCTL parameters mentioned in this table. See “Defining the IMS DRA startup parameter table” on page 27 for information about DRA startup table parameters.

Table 1. CICS system initialization parameters and DBCTL

System initialization parameter	D	R	IMS/DBCTL startup parameter	Comments
APPLID	Y	Y	N/A	The generic z/OS Communications Server application identifier for this CICS system.
DBCTLCON	Y	N	N/A	YES specifies that you want CICS to connect to a DBCTL subsystem automatically during CICS initialization. This causes CICS to invoke the DBCTL attach program, DFHDBCON. The other information CICS needs for starting the attachment, such as the DRA startup table suffix or the DBCTL subsystem name, is taken from an **INITPARM** system initialization parameter. Specifying DBCTLCON=YES means you do not have to define the DBCTL attach program in the CICS post-initialization program list table (PLT), as described in “Program list table (PLT)” on page 16.
DSALIM	Y	Y	N/A	Upper limit of the total amount of storage within which CICS can allocate the individual dynamic storage areas (DSAs) below the 16 MB line. See DSALIM system initialization parameter in the System Definition Guide for information about specifying DSALIM. See *IMS System Administration Guide* for guidance on DBCTL storage estimates.
EDSALIM	Y	Y	N/A	Upper limit of the total amount of storage within which CICS can allocate the individual dynamic storage areas (EDSAs) above the 16 MB line. For more information, see EDSALIM system initialization parameter in the System Definition Guide. See *IMS System Administration Guide* for guidance on DBCTL storage estimates.
INITPARM	Y	N	N/A	Used to pass parameters to programs (for example, PLT programs) during CICS startup. With DBCTL, you can use it to specify DRA startup parameter table suffix and DBCTL identifier to automate connection to a particular DBCTL. **INITPARM** applies to COLD, INITIAL, WARM, or EMERGENCY starts of CICS.
PDIR	N	Y	N/A: use APPLCTN	Suffix of the PDIR. With DBCTL, the PDIR is generated during DBCTL generation using the APPLCTN macro.
PSBCHK	Y	Y	N/A	Requests PSB authorization checking of a remote terminal initiating a transaction using transaction routing. To obtain the check, you must also specify YES or name on the **XPSB** system initialization parameter.

Table 1. CICS system initialization parameters and DBCTL (continued)

System initialization parameter	D	R	IMS/DBCTL startup parameter	Comments
XPSB	Y	Y	N/A	Security class name by which PSBs are defined to RACF®. For DBCTL, you specify the RACF resource class to be used to security check PSBs. For more information, see CICS resource class system initialization parameters in the RACF Security Guide.

PSB directories (PDIRs)

PSB directories (PDIRs) contain entries defining each PSB to be accessed using remote DL/I.

If you are using DBCTL exclusively, you do not need to generate a PDIR for CICS. Instead you must define PSBs and DMBs using the IMS macros APPLCTN and DATABASE respectively. (For information on the APPLCTN and DATABASE macros, see "Generating DBCTL" on page 17.)

If you want to function ship requests to a CICS system, at which the database manager may be DBCTL or remote DL/I (function shipping), you will need to generate a PDIR.

CICS routes DL/I requests to remote DL/I or DBCTL according to the PSB that is named. If the PSB appears in the CICS PDIR, the request is routed to remote DL/I (that is, function shipped to another CICS system). If the PSB does not appear in the CICS PDIR, and CICS is connected to DBCTL, CICS routes the request to DBCTL. In addition, if the PSB appears in the PDIR and specifies a SYSID that matches the local SYSID, the request is routed to DBCTL.

DD statements

You must put these two modules, which appear in the IMS.RESLIB library, in the CICS STEPLIB data set concatenation.

- The DRA startup parameter table: DFSPZPxx (where xx is the user-defined suffix)
- The DRA startup router program: DFSPRRC0.

You can do this by placing a DD statement for IMS.RESLIB in the CICS STEPLIB concatenation (which must be APF-authorized). For example:

```
//STEPLIB   DD  DSN=CICSTS42.CICS.SDFHAUTH,DISP=SHR
//          DD  DSN=IMS.RESLIB,DISP=SHR
```

IMS.RESLIB (which must also be APF-authorized) contains a default DRA startup table, in which the suffix is set to 00. You can generate your own versions into this library. If you decide to use a ***different*** library for your own versions, make sure it is APF-authorized, and is included in the CICS STEPLIB concatenation.

The DRA will dynamically allocate the IMS.RESLIB library using the DD name CCTLDD and the data set name IMS.RESLIB, unless either has been overridden in the DRA startup parameter table.

DD statements removed from CICS JCL in a DBCTL-exclusive environment

The following DD statements are not required in a DBCTL environment.

DFSCTL
For DBCTL, DFSCTL is not required. DBCTL owns the OSAM buffer pools, which are specified in DBCTL startup JCL and in the DRA startup parameter table. See "Database buffer specifications and option parameters" on page 25 and "Defining the IMS DRA startup parameter table" on page 27.

DFSRESLB
For DBCTL, DFSRESLB is not required. DFSRESLB is replaced by the DRA dynamically allocating IMS.RESLIB as described in "DD statements" on page 14.

IEFRDER
Used to define DL/I batch logging. For DBCTL, DL/I logging is to the IMS log. See "Defining IMS logging parameters" on page 24.

IMSMON
With DBCTL, you can start and stop the IMS monitor dynamically. See "Using the IMS monitor" on page 150.

IMSACB
For DBCTL, IMSACB is in the DBC procedure and the DLS procedure. There are additional DD statements: IMSACBA and IMSACBB. One is the active library and the other is available for the IMS online change utility.

DFSVSAMP
For DBCTL, DFSVSAMP is not used. The information it contains, for example, VSAM buffer parameters and performance and trace options, is in the DFSVSMxx member of IMS.PROCLIB in the PROCLIB DD statement of the DBCTL startup procedure (DBC). The DFSVSMxx member must be available to DLISAS, which means that you must add a data set with member DFSVSMxx to the DLISAS address space. The last two characters of the DFSVSM member are a suffix, which you specify in the VSPEC parameter of the DBCTL startup procedure (DBC).

RECON data sets
RECON data sets are generally specified in DFSMDA IMS dynamic allocation members in the IMS.RESLIB library. See "IMS dynamic allocation macro (DFSMDA)" on page 25. For DBCTL, RECON data sets can be specified in the DBRC procedure.

JCLPDS
For DBCTL, JCLPDS is in the DBRC procedure.

JCLOUT
For DBCTL, JCLOUT is in the DBRC procedure.

Database DD statements
Generally, you specify database DD statements in DFSMDA IMS dynamic allocation members in the IMS.RESLIB library. For DBCTL, they can be specified in the DLS address space for DL/I databases, or in the DBC address space for DEDBs.

CICS-supplied groups within CICS system definition

Program, transaction, and mapset entries for the CICS system definition (CSD) file to provide DBCTL support are supplied in the group DFHDBCTL.

This includes the DBCTL connection and disconnection transaction, CDBC, the inquiry transaction, CDBI, and the operator transaction, CDBM. DFHDBCTL is in DFHLIST, which contains the CICS resource definitions needed to run IBM® supplied transactions that must be installed in your system. Also in DFHLIST is the DFHEDP group, which provides the program definition required to run EXEC DLI applications. The group DFHEDP must always be installed in the CICS system. If you need further information about DFHLIST, see "CICS-supplied resource definitions, groups, and lists" in the *CICS Resource Definition Guide*.

You might also want to specify the following options of the TRANSACTION definition for transactions using DBCTL:

- RESTART

  This option defines whether CICS will attempt to restart a transaction that has been backed out after a failure. (See “Deadlocks and interactions with automatic restart” on page 81.)
- SPURGE

  Specify SPURGE(YES) so that the transaction can be purged using CEMT. “Purging a transaction that is using DBCTL” on page 60 tells you how to use CEMT in this way.

Log management

All DBCTL-related information is sent to the IMS log, not the CICS system log.

This method of logging uses the IMS log utilities and the online log data sets (OLDS) and write-ahead data sets (WADS). Because database change records are written to the IMS log, you do not need to retain the CICS system log for use by IMS database recovery utilities in a DBCTL-exclusive environment. IMS logging operations are described in “IMS logging” on page 22.

Monitoring control table (MCT)

If you were using local DL/I when converting to DBCTL, you can remove the entries for the DL/I event monitoring points (EMPs) from the monitoring control table (MCT).

However, you will need additional monitoring control table (MCT) entries if you want to provide support for the monitoring information returned from DBCTL. These MCT entries are in CICSTS42.CICS.SDFHSAMP in the copy member DFH$MCTD.

Program list table (PLT)

To connect CICS to DBCTL at CICS startup time, you can invoke it in the second stage of program list table postinitialization (PLTPI) processing (that is, the third stage of CICS initialization).

You do this by including an entry for DFHDBCON (the DBCTL connection program) using the DFHPLT macro. Including an entry for DFHDBCON in the PLT enables you to connect automatically to the same DBCTL as when the system was last shut down, or to a different one. For more information, see “Connecting DBCTL to CICS automatically” on page 36.

As an alternative, you can use the **DBCTLCON** system initialization parameter to make the automatic connection. For more information, see Table 1 on page 13.

Transient data queues

You will need a definition for the CDBC transient data queue. The CDBC transient data queue is used for messages issued by the CICS-DBCTL interface.

You can suppress or reroute messages sent to transient data queues such as CDBC. You can reroute from CDBC to a list of consoles, from CDBC to a different transient data queue, or reroute console messages to CDBC. For programming information about coding the CICS-supplied user exit used to re-route messages, and on the example user exit provided to help you do so, see *CICS Customization Guide*.

Generating DBCTL

You generate the appropriate IMS control blocks and resource definitions for a DBCTL subsystem by performing an IMS system definition.

About this task

IMS system definition is a two-stage process with an optional preprocessor. Stage 1 checks your input specifications (appropriate JCL and macro statements, which are described below) and generates a series of MVS job steps for stage 2. Stage 2 builds IMS system libraries, execution procedures, and the DBCTL control program. The optional preprocessor is a convenient tool that checks for duplicate names and checks the length and format of the names used as input for stage 1.

This topic covers:

Defining the DBCTL subsystem

IMS uses macro statements for system definition. These macro statements define the operating systems, operating system interfaces, storage pools, PSBs, and databases. From some of these macro statements, DBCTL constructs a set of control blocks with which to execute.

About this task

To define the environment in which DBCTL operates, you use DBCTL startup parameters and control information in a number of IMS system data sets. You then use the appropriate suffixes to specify the information to be used for a particular DBCTL run. This is like selecting CICS tables by specifying their suffixes in the SIT or in SIT overrides.

The IMS system generation macros you need are listed in "IMS system generation macros used by DBCTL" on page 18. See *IMS System Definition Reference* or *IMS Installation Volume 2: System Definition and Tailoring* for guidance on the syntax of these macros. "Illustration of DBCTL startup parameter creation and selection" on page 32 shows how DBCTL startup parameters are created and selected during startup. If you are new to IMS system definition, you might find it helpful to look at this illustration while reading the information about generating DBCTL.

IMS system generation macros used by DBCTL

DBCTL uses the IMSCTRL, MAXREGN, APPLCTN, BUFPOOLS, DATABASE, FPCTRL, IMSCTF, SECURITY, and IMSGEN macros.

- IMSCTRL

  The first macro in a DBCTL system generation is IMSCTRL. It is always required and there can be only one within each IMS system definition. IMSCTRL describes the MVS system under which IMS executes, the type of IMS system, the type of generation to be performed, and the components of the IMS environment, for example, IRLM and DBRC. Because DBRC is mandatory for DBCTL, you do not need to specify the IMSCTRL parameter, DBRC=YES. (If you do specify this parameter, it is ignored.) You can use IMSCTRL to cause the IMS nucleus and the DDIR and PDIR to be regenerated.

  – MAXREGN

    MAXREGN is the number of regions (threads) that DBCTL allocates at startup. MAXREGN takes a value 1 - 999. It can increase dynamically to a maximum of 999. Each BMP needs one region. Each connected CICS needs from MINTHRD to MAXTHRD regions. See also MINTHRD and MAXTHRD, which are used to specify the minimum and maximum numbers of threads for a particular CICS system, as described in "Defining the IMS DRA startup parameter table" on page 27. For information about how these parameters interact, see "Specifying numbers of threads" on page 154. (MAXREGN is not the only parameter you need in IMSCTRL, but is mentioned here to contrast it with MINTHRD and MAXTHRD.)

- APPLCTN

  You use the APPLCTN macro to name PSBs (one macro for each PSB) that are to be used by application programs to access databases through DBCTL.

  If multiple CICS transactions or BMPs are to schedule a PSB concurrently, the APPLCTN macro for that PSB must specify SCHDTYP=PARALLEL. ***If you do not specify SCHDTYP=PARALLEL, only one transaction at a time is able to schedule a PSB***. You can change the SCHDTYP of a PSB using the online change process and the /MODIFY command, which you enter at the DBCTL console. See "Changing DBCTL resources online" on page 57 for more information about the online change process and the /MODIFY command.

  In DBCTL, PSBs used by CICS transactions can be defined either with the TP option or the BATCH option. In the example in Figure 5 on page 21, the BATCH option is used. Figure 5 on page 21 also includes an example of defining a PSB for the CDBM operator transaction.

- BUFPOOLS

  You use the BUFPOOLS macro to specify default main storage buffer pool sizes for DBCTL, including the size of the DMB and PSB pools. You can override these values at startup using the CSAPSB=, DLIPSB=, and DMB= parameters.

- DATABASE

  You use DATABASE macro statements to define the databases that DBCTL accesses (one macro for each database). Each physical database must be referenced on a DATABASE macro statement. You can change this resource through the online change process using the /MODIFY command, which you enter at the DBCTL console. See "Changing DBCTL resources online" on page 57 for more information about the /MODIFY command.

- FPCTRL

  The FPCTRL macro statement defines the fast path options when DEDBs are used. You must use this macro only if you want DEDB support.

Note: For DBCTL users, fast path support refers only to DEDBs. Parameters that begin with FP refer to DEDBs in a DBCTL-exclusive environment.

- IMSCTF

  The IMSCTF macro statement includes parameters to define the SVCs to be used by DBCTL, logging options, and the device type for DBCTL's restart data set.
- SECURITY

  The SECURITY macro statement enables you to specify optional security features to be in effect during IMS execution, unless they are overridden during system initialization.

  If you are implementing IMS security use the Resource Access Control Facility (RACF), see *IMS Administration Guide: System* or *IMS System Administration Guide*. For more information about security with DBCTL, see Chapter 6, "Security checking with DBCTL," on page 117.
- IMSGEN

  The IMSGEN macro statement must be the last system definition macro in the Stage 1 input. It specifies the assembler and linkage editor data sets and options, and the system definition output options and features. It specifies the suffix character for the IMS nucleus (DFSVNUCx in IMS.RESLIB) and for the DDIR (DFSDDIRx) and PDIR (DFSPDIRx) in IMS.MODBLKS. You must specify the MACLIB parameter of the IMSGEN macro as MACLIB=ALL when using DBCTL for the first time.

Implementing CICS-supplied transaction CDBM

CICS provides a transaction, CDBM, that enables DBCTL operator commands to be input from a CICS terminal. The CICS terminal must be a BMS supported device.

About this task

"CDBM operator transaction" on page 47 has more information about CDBM. To use CDBM, you must have a DBCTL system running IMS.

CDBM uses the AOI commands that can be issued across the DRA interface between CICS and DBCTL. For more information about these commands, see "Issue IMS AIB call format" on page 86.

Choose either of these methods to implement CDBM:

Procedure

1. Use PSBGEN to generate, and add to the DBCTL system, a PSB named DFHDBMP.
   a. Specify parallel scheduling for DFHDBMP, so that multiple CDBM transactions can be active at the same time.
   b. DFHDBMP does not need to have any associated PCBs.
   c. The IOASIZE parameter must be large enough to cope with the largest AOI command issued. Large AOI commands can result from using wild cards. For example, issuing CDBM /START DATABASE D* results in a start command being issued for all database names beginning with D. See *IMS Utilities Reference: Systems* for information on defining IOASIZE.

   Example input for PSBGEN is:

   ```
   PSBGEN LANG=ASSEM,PSBNAME=DFHDBMP,IOASIZE=1000
   ```

2. Alternatively, with IMS V10, you can use the batch SPOC (Single Point of Control) interface to create DFHDBMP. Specify the following command in the batch SPOC:

```
CREATE PGM NAME(DFHDBMP) SET( BMPTYPE(Y) DOPT(N) +
FP(N) GPSB(Y) LANG(ASSEM) RESIDENT(N) +
SCHDTYPE(PARALLEL) TRANSTAT(N))
```

Modifying IMS system data sets using online change

You can modify the IMS system data sets MODBLKS, MATRIX, and ACBLIB using online change.

About this task

Each of them must be present in the following copies:

- A staging library, which is identified by an unsuffixed DD statement (MODBLKS, MATRIX, ACBLIB), and is used offline only to prepare changes to the active library.
- An active and an inactive library, which are used in flip-flop mode and are identified by suffixed DD statements (MODBLKSA and MODBLKSB, and so on). The same parameter (MODBLKSx, where x= A or B) controls the active library for both MODBLKS and MATRIX. While the active library (either ...A or ...B) is being used online by DBCTL, you can use the online change utility to copy the contents of the staging library to the inactive library. You use a series of /MODIFY commands to perform the actual switch from the active library to the updated inactive library.

The IMS.MODSTAT data set, which is created during the IMS system generation and updated automatically, indicates which of the suffixed data sets is currently active. For guidance on using online change, see “Changing DBCTL resources online” on page 57 and *IMS System Administration Guide* or *IMS Administration Guide: System*.

Example of JCL required to generate a basic DBCTL subsystem

You can copy and modify this JCL example to generate a DBCTL subsystem.

The minimum generation required to generate DBCTL is ON-LINE,DBCTL. (You must perform an online generation to change the SVC numbers.) You must include the dash (-) in the ON-LINE parameter. If you do not, you get the following messages when you try to generate DBCTL:

```
** ASMA254I *** MNOTE ***    76+     4,G002 FOLLOWING OPERAND(S) OMITTED OR INVALID:
** ASMA254I *** MNOTE ***    77+     4,      SYSTEM
```

You use an ACB generation to create members of the IMS.ACBLIB. See *IMS Utilities Reference: Database* for further guidance on doing this.

Figure 5 on page 21 shows an example DBCTL generation that you can copy and modify to generate a DBCTL subsystem. This example includes only the parameters needed to get a “basic” system up and running. This example does not include optional parameters, such as those for DEDB support, and it assumes that you want to tune other parameters (such as the number of threads) later, when you have had an opportunity to see how the subsystem runs.

Note: You can, instead, use the IMS INSTALL/IVP dialog to generate stage 1 macros for DBCTL. For guidance on doing so, see *IMS Installation Guide*.

```
//DBCGEN   JOB 1,PGMERID,
//         MSGCLASS=A,MSGLEVEL=(1,1),
//         CLASS=A,NOTIFY=PGMERID
//ASM EXEC PGM=ASMA90,
// PARM='DECK,NOOBJECT',
//       REGION=4096K
//SYSLIB DD DSN=IMS.OPTIONS,DISP=SHR
//       DD DSN=IMS.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MACLIB,DISP=SHR
//*
//SYSUT1 DD UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSUT2 DD UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSUT3 DD UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSPRINT DD SYSOUT=*
//SYSPUNCH DD DSN=IMS.STAGE2,DISP=SHR
//SYSIN    DD *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
*  SAMPLE DBCTL SYSTEM DEFINITION STAGE 1 INPUT SPECIFICATIONS        *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         IMSCTRL SYSTEM=(VS/2,(ON-LINE,DBCTL),3.1),                    X
               MAXREGN=(20,52K,A,A),                                   X
               MCS=(2,7),DESC=7,MAXCLAS=1,IMSID=IMSA
*
         IMSCTF SVCNO=(,203,202),                                      X
               LOG=(DUAL,MONITOR),                                     X
               RDS=(3380,4096),                                        X
               CPLOG=1000,CORE=(,50,1)
*
*        DEFINE SYSTEM BUFFERS
*
         BUFPOOLS PSBW=60000,DMB=10000,SASPSB=(20000,80000)
*
*        DEFINE DL/I DATABASES
*
         DATABASE RESIDENT,DBD=DI21PART
```

Figure 5. Example JCL to generate DBCTL 1/2

```
*        DEFINE SAMPLE APPLICATIONS
*
         APPLCTN PSB=DFHSAM04,PGMTYPE=BATCH,SCHDTYP=PARALLEL
         APPLCTN PSB=DFHSAM05,PGMTYPE=BATCH,SCHDTYP=PARALLEL
         APPLCTN PSB=DFHSAM14,PGMTYPE=BATCH,SCHDTYP=PARALLEL
         APPLCTN PSB=DFHSAM15,PGMTYPE=BATCH,SCHDTYP=PARALLEL
         APPLCTN PSB=DFHSAM24,PGMTYPE=BATCH,SCHDTYP=PARALLEL
         APPLCTN PSB=DFHSAM25,PGMTYPE=BATCH,SCHDTYP=PARALLEL
         APPLCTN PSB=DFHDBMP,PGMTYPE=BATCH,SCHDTYP=PARALLEL
*
         IMSGEN ASM=(H,SYSLIN),                                        X
               ASMPRT=ON,                                              X
               LKPRT=(XREF,LIST),                                      X
               LKSIZE=(880K,64K),                                      X
               LKRGN=4096K,                                            X
               SUFFIX=1,                                               X
               SURVEY=NO,                                              X
               SYSMSG=TIMESTAMP,                                       X
               MACLIB=ALL,                                             X
               OBJDSET=IMS.OBJDSET,                                    X
               USERLIB=IMS.LOADLIB,                                    X
               PROCLIB=(YES,),                                         X
               NODE=(IMS,IMS,IMS),                                     X
               JCL=(GENJOB,                                            X
               (1),                                                    X
               PGMERID,                                                X
               A,                                                      X
               (TIME=5,CLASS=K,NOTIFY=PGMERID)),                       X
               SCL=(99)
         END
```

Figure 6. Example JCL to generate DBCTL 2/2

For more detailed system definition examples and further guidance on selecting the appropriate system definitions, and for IMS system definition examples, see *IMS System Definition Reference* or *IMS Installation Volume 2: System Definition and Tailoring*.

IMS logging

IMS logging uses two types of data set: online log data sets (OLDS) and write ahead data sets (WADS).

These data sets are described below. For further guidance on using the OLDS and the WADS, see *IMS Operations Guide*.

IMS online log data set (OLDS)

IMS writes log records to a DASD data set called the online log data set (OLDS).

The OLDS is made up of multiple data sets written in wraparound form. Using more than one OLDS enables IMS to continue logging when the first OLDS is full. Also, if an I/O error occurs while writing to an OLDS, IMS can continue logging by isolating the OLDS where the problem occurred and switching to another one.

IMS can write committed log records to the write-ahead data set (WADS) so that these records are externalized to avoid the need to write partially filled and padded log blocks to the OLDS. The WADS is described in "IMS write-ahead data set (WADS)" on page 23.

When the OLDS is full, it is archived to the system log data set (SLDS). How frequently the OLDS is archived depends on whether you specified automatic archiving using the ARC=parameter in the DBC JCL. You can specify ARC=1

through ARC=99. Automatic archiving takes place only when the number of OLDS you specified is full. The system reuses the OLDS after it has been archived. An SLDS can be on DASD or on tape. The contents are used as input to the database recovery process.

IMS archives the OLDS using the log archive utility (DFSUARC0). During archiving, IMS can write a subset of the log records it writes to the SLDS to the recovery log data set (RLDS). This subset consists only of the log records required to perform a database recovery.

During logging, IMS writes system checkpoint ID information (including OLDS positioning information) to the restart data set (RDS). IMS uses the RDS during the restart process to determine from which checkpoint to begin a restart. (See *IMS Operations Guide* for further guidance about the RDS.)

IMS write-ahead data set (WADS)

The main purpose of the write-ahead data set (WADS) is to contain a copy of committed log records that are in the OLDS buffers, but have not yet been written to the OLDS because the OLDS buffer is not yet full.

IMS uses the WADS to avoid the need to write partially filled and padded blocks to the OLDS. WADS space is continually reused after the appropriate log data has been written to the OLDS. If there is a system failure, IMS uses the log data in the WADS to complete the content of the OLDS in use, and then closes the OLDS as part of an emergency restart. This is also an option of the IMS log recovery utility (DFSULTR0). (The OLDS must be closed before database recovery can take place.) You can change the following specifications for the WADS at any restart:

- Number of WADSs
- Sequence of WADSs
- WADSs data set names
- Use of single or dual WADSs.

Log control with DBRC

Database Recovery Control (DBRC) assists you in controlling DBCTL logs and in managing recovery of databases. With DBCTL, you ***must*** use DBRC to control DBCTL logs, and you may optionally use it to control batch logs and database recovery.

DBRC places the information it uses to control recovery in the RECON data sets, which are required with DBCTL. These data sets include information about the OLDS; for example, it indicates whether an OLDS is available for use or contains data that must be archived.

Define three RECON data sets when you install DBRC. Two of the RECON data sets are active; the third is a spare. For most purposes, you can think of the two active RECON data sets as a single RECON data set, or the RECON.

DBCTL requires DBRC to be at SHARECTL level; if it is not, DBCTL will not start. To initialize the RECON specify (or let it default to) INIT.RECON SHARECTL. Figure 7 on page 24 shows some example JCL you can copy to initialize the RECON.

```
//INITREC JOB 1,PGMERID,CLASS=Q,MSGCLASS=A
//*
//RECON    EXEC PGM=DSPURX00,REGION=1000K
//STEPLIB  DD DSN=IMS.RESLIB,DISP=SHR
//DFSRESLB DD DSN=IMS.RESLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//RECON1   DD DSN=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//SYSIN    DD *
 INIT.RECON SSID(IMSA)
/*
```

Figure 7. Example JCL to initialize the RECON

If you already have a RECON, specify (or let it default to) CHANGE.RECON SHARECTL. When the OLDS is full, DBRC starts a log archive job. Skeleton JCL statements are edited by DBRC before the job is submitted. The skeleton JCL is member ARCHJCL of the library specified in the JCLPDS DD statement in the DBRC JCL. You do not have to wait for the OLDS to fill in order to test the automatic log archive. Instead, you can cause the OLDS to switch using the DBCTL operator command /SWITCH OLDS. Alternatively, you can use the /DBRECOVERY without the NOFEOV keyword. For guidance on the syntax of the /SWITCH and /DBRECOVERY commands, see *IMS Operator's Reference* . (See also "Operator communication with DBCTL: overview" on page 43 for information on using DBCTL operator commands.)

For detailed guidance on automatic log archiving and DBRC skeleton JCL, see *IMS Utilities Reference: Database* . For further guidance on using DBRC, see *IMS Operations Guide*.

Defining IMS logging parameters

You define IMS logging parameters in member DFSVSMxx in the IMS.PROCLIB, identified by DD name PROCLIB in the DBC and DLISAS JCL.

About this task

You specify the suffix xx for DFSVSMxx in the DBCTL startup parameter VSPEC. For an illustration of the parameters involved, see "Illustration of DBCTL startup parameter creation and selection" on page 32. The logging parameters in DFSVSMxx include:

- Number of OLDS
- Number of OLDS buffers
- Selection of single or dual OLDS
- Number of WADS.

A further logging parameter, used to specify single or dual copies of the WADS is in the DBCTL startup parameters. See "Starting DBCTL, DLISAS, and DBRC" on page 26 for information about the DBCTL startup procedure.

You must preallocate the OLDS and WADS data sets and specify the block size when the data set is allocated. See *IMS Installation Guide* for guidance on doing this.

Provide dynamic allocation members for all OLDS and WADS data sets. See "IMS dynamic allocation macro (DFSMDA)" on page 25.

Archiving

DBRC automatically submits a job to archive the OLDS when:

- IMS terminates
- The OLDS fills and logging switches to an empty OLDS
- You issue a /DBRECOVERY command without the NOFEOV keyword
- You switch the OLDS manually.

See *IMS Operations Guide* and *IMS Utilities Reference: Database* for guidance on implementing automatic archiving, and *IMS Operator's Reference* for the syntax of the /DBRECOVERY command. (You can also use the /DBRECOVERY command without the NOFEOV keyword to test your implementation.)

IMS dynamic allocation macro (DFSMDA)

Use the IMS dynamic allocation macro (DFSMDA) in all production databases, because:

- Allocation is controlled from a central point.
- You do not have to change DBCTL JCL or batch job JCL in order to change a data set name.
- It avoids possible confusion over which DBCTL address space requires the DD statement for a database, because the library with the DFSMDA members can be concatenated in the STEPLIB DD statement.
- If you do not use DFSMDA, DL/I database DD statements must be in the DLISAS (DLS) address space, and DEDB DD statements must be in the DBCTL (DBC) address space.

To use dynamic allocation, you need one member per database in the IMS.RESLIB library (or an authorized STEPLIB library), using the IMSDALOC procedure to assemble and link-edit the appropriate DFSMDA macros. See *IMS System Administration Guide* or *IMS Administration Guide: System* for general guidance on dynamic allocation and *IMS Utilities Reference: Database* for guidance on using the DFSMDA macro.

Database buffer specifications and option parameters

You define the VSAM and OSAM database buffer pool specifications and IMS performance and trace options in the DFSVSMxx member of the IMS.PROCLIB data set, which is pointed to by the PROCLIB DD statement of the DBCTL startup procedure (DBC).

The last two characters of the DFSVSMxx member are a suffix. You specify this suffix in the VSPEC parameter of the DBCTL startup procedure. See *IMS System Definition Reference* for guidance on the syntax of these parameters and *IMS Database Administration Guide* for guidance on specifying the database buffer pool parameters. For an illustration of the parameters involved in DBCTL startup, see "Illustration of DBCTL startup parameter creation and selection" on page 32.

Overriding DBCTL generation parameters at execution time

About this task

You can change many IMS system definition values at DBCTL startup using parameters on the DBC procedure. You can specify these override parameters on the PARM of the EXEC statement. However, there is a 100-character limit to the length of the PARM field you can specify on a JCL EXEC statement, which means

that you cannot override all possible DBC parameters in the JCL. A better approach is to use member DFSPBDBC, which allows you to specify DBCTL control region execution parameters that override those specified in the stage 1 macros.You can place several DFSPBDBC members in PROCLIB by replacing the member name DFSPBDBC with DFSPBxxx, where *xxx*must be three alphanumeric characters. The RGSUF= keyword in the DBC procedure specifies the *xxx* suffix to be used during startup of the DBCTL control region. For more information about DFSPBDBC, see *IMS Installation Volume 2: System Definition and Tailoring*.

Naming convention

The DBCTL display commands (for example, /DISPLAY ACTIVE and /DISPLAY CCTL, described in "Finding out current status of DBCTL activities" on page 55). and the DRA startup table USERID parameter, all use what is known in IMS and DBCTL as the CCTL ID to identify the transaction management subsystem. In the case of CICS, the CCTL is CICS and the ID is the CICS APPLID.

However, many IMS messages use the jobname of the CICS system instead. An example of this sort of message is DFS554, which notifies you that a BMP region, or a thread from a CICS transaction, has terminated abnormally. If the DFS554 message was caused by an abnormal termination of a thread that originated from ***CICS***, the message text contains the CICS job name or CICS startup procedure name. You will therefore need a naming convention that enables operators to immediately identify a corresponding CICS APPLID and CICS JOBNAME. For example, if you use the APPLID DBDCCICA, your job name could also contain the characters CICA.

Starting DBCTL, DLISAS, and DBRC

You use the procedure library member DBC that is supplied with DBCTL to start the DBCTL subsystem.

About this task

The procedure is generated during IMS system definition and must be modified to fit your system's needs.

Also generated during system definition are procedures for DBRC and DLISAS, which are used to generate the DBRC and DLISAS address spaces. The DBRC and DLISAS procedures are started automatically by DBCTL during DBCTL startup.

The region types specified for each one are:

```
PARM='DBC'
for DBCTL PARM='DRC' for DBRC PARM='DLS' for DLISAS
```

All three procedures use positional parameters on the EXEC statement:

```
PARM='region type,parm1,parm2,parm3,...'
```

Many of the positional parameter defaults are specified during system generation, but you can override them with parameters you specify at execution time.

When all three address spaces have been started successfully, DBCTL issues the following message indicating it is ready to accept an appropriate restart command:

```
DFS989I IMS (DBCTL) READY (CRC=x) xxxx
```

where x is the command recognition character (CRC), as explained in "Operator communication with DBCTL: overview" on page 43, and xxxx is the DBCTL sysid, as specified in the IMSID= parameter of the DBCTL startup JCL.

See *IMS System Definition Reference* or *IMS Installation Volume 2: System Definition and Tailoring* for guidance on DBCTL procedures, including JCL and descriptions of parameters.

Defining the IMS DRA startup parameter table

The DRA startup parameter table provides the parameters needed to define the interface to the DBCTL subsystem.

About this task

You create the DRA startup parameter table by assembling the DFSPRP macro and link-editing it into the IMS.RESLIB library (or another APF-authorized library) as DFSPZPxx, where xx=00, for the default, or any other alphanumeric characters. Unless your IMS RESLIB uses the default name IMS.RESLIB, supplied in DFSPZP00, you must specify the name you have chosen in your version of the DRA. In "Example JCL to generate a DRA startup table" on page 29, the name IMS.RESLIB is used.

Note: The macro used is DFSPRP, but the name of the module you must link edit is DFSPZPxx. You must also link edit the DRA into an authorized library that is part of the CICS STEPLIB concatenation.

The DFSPRP macro has the following parameters:

- DSECT=NO

  A DSECT statement for PZP is not generated. You must specify this option in order to create a CSECT, which is required in order to assemble the module DFSPZPxx.
- FUNCLV=

  The CCTL (in this case, CICS) functional level. The default (and the only valid value) is 1.
- DDNAME=

  A 1- to 8-character ddname to be used with dynamic allocation of the DRA RESLIB. The default is CCTLDD.
- DSNAME=

  A 1- to 44-character data set name of the DRA RESLIB. The default is IMS.RESLIB.
- DBCTLID=xxxx

  The 1- to 4-character name of the DBCTL address space. The default is SYS1. This parameter must be the same as the IMSID in the DBCTL startup procedure for the DBCTL to which you want this CICS to connect. You can connect multiple CICS systems to the same DBCTL, but a CICS system can connect to only one DBCTL at a time.
- USERID=xxxxxxxx

  CICS users do not specify this parameter; it is supplied by CICS itself. If you do specify anything, CICS overrides it. USERID is the 1- to 8-character name of the CICS address space (or CCTLID). The value CICS supplies when it connects to DBCTL is the CICS APPLID.
- MINTHRD=xxx

This parameter specifies the number of threads for this CICS system that, once initialized, remain created while the DRA is active. These threads remain allocated until this CICS system is disconnected from DBCTL, except if a thread is stopped by a /STOP command or by a thread failure. Additional threads are created, up to the number specified in MAXTHRD, or the number specified in MAXREGN, or the maximum of 999, whichever of these values is the lowest. These additional threads (not the MINTHRDs) are released when there is not enough system activity to require them. The maximum value you can specify for MINTHRD is 999, and the default is 1. For information about specifying values for MINTHRD, see "Specifying numbers of threads" on page 154. See also MAXREGN in "IMS system generation macros used by DBCTL" on page 18.

- MAXTHRD=xxx

  This parameter specifies the maximum number of transactions for which this CICS system can have PSBs scheduled in DBCTL. Any schedule requests that are over this limit are queued in the DRA. You can balance the load sent to a single DBCTL from multiple CICS systems by specifying appropriate values for MAXTHRD in each CICS.

  The maximum value you can specify for MAXTHRD is 999 (but it should not exceed the value specified for MAXREGN) and the default is 1, or the value you specified in MINTHRD. For information about specifying values for MAXTHRD, see "Specifying numbers of threads" on page 154. See also MAXREGN in "IMS system generation macros used by DBCTL" on page 18.

- TIMER=xx

  The frequency, in seconds, with which CICS is to repeat attempts to connect to DBCTL when connection has failed and the console operator has requested that CICS wait for connection in reply to a DFS690 message (rather than canceling the connection attempt). You can specify any value from 0 through 99. However, note that if you specify 0, the default value is used. The default is 60.

- CNBA=xxx

  The total number of DEDB buffers that are allocated for this CICS system. The default is 0.

- FPBUF=xxx

  The number of DEDB buffers to be allocated and fixed per thread. The default is 0. See "DEDB performance and tuning considerations" on page 155 for information about defining DEDB buffer pools.

- FPBOF=xxx

  The number of DEDB overflow buffers to be allocated per thread. The default is 0. See "DEDB performance and tuning considerations" on page 155 for information defining DEDB buffer pools.

  Notes:

  1. For DBCTL users, fast path support refers only to DEDBs. Parameters that begin with FP refer to DEDBs in the DRA startup table.
  2. You do not need the parameters CNBA, FPBUF, and FPBOF if you are not using DEDBs.
  3. For detailed guidance on specifying DEDB buffers, see *IMS System Administration Guide* or *IMS Administration Guide: System*.

- TIMEOUT=xxx

  The amount of time, in seconds, that CICS should wait for a DRA TERM request to complete. The maximum value is 999, and the default is 60. For guidance on what to specify, see TIMEOUT in "CICS failure" on page 78.

- SOD=x

The output class to be used for a snap memory dump of abnormal thread terminations. The default is A. See "Dumps produced by the DRA" on page 136 for more information about these memory dumps.

- AGN=xxxxxxxx

  The 1- to 8-character application group name (AGN). You must use this parameter only if you have specified AGN security checking for DBCTL. There is no default. See Chapter 6, "Security checking with DBCTL," on page 117 for more information.

Example JCL to generate a DRA startup table

Some example JCL you can copy to generate a DRA.

```
//DRAJOB   JOB 1,PGMERID,MSGCLASS=A,MSGLEVEL=(1,1),
//         CLASS=A,NOTIFY=PGMERID
//ASM EXEC PGM=ASMA90,
//       PARM='DECK,NOOBJECT,LIST,XREF(SHORT),ALIGN',
//       REGION=4096K
//SYSLIB DD DSN=IMS.OPTIONS,DISP=SHR
//       DD DSN=IMS.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MACLIB,DISP=SHR
//*
//SYSUT1 DD UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSUT2 DD UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSUT3 DD UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSPUNCH DD DSN=&&OBJMOD,
//       DISP=(,PASS),UNIT=SYSDA,
//       DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),
//       SPACE=(400,(100,100))
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
PZP      TITLE 'DATABASE RESOURCE ADAPTER STARTUP PARAMETER TABLE'
DFSPZP00 CSECT
*********************************************************************
*        MODULE NAME: DFSPZP00                                      *
*                                                                   *
*        DESCRIPTIVE NAME: DATABASE RESOURCE ADAPTER (DRA)          *
*                  STARTUP PARAMETER TABLE.                         *
*                                                                   *
*        FUNCTION: TO PROVIDE THE VARIOUS DEFINITIONAL PARAMETERS   *
*                  FOR THE COORDINATOR CONTROL REGION. THIS         *
*                  MODULE MAY BE ASSEMBLED BY A USER SPECIFYING     *
*                  THEIR PARTICULAR NAMES, ETC. AND LINKEDITED      *
*                  INTO THE USER RESLIB AS DFSPZPXX.  WHERE XX      *
*                  IS EITHER 00 FOR THE DEFAULT, OR ANY OTHER ALPHA- *
*                  NUMERIC CHARACTERS.                              *
*                                                                   *
*********************************************************************
         EJECT
         DFSPRP DSECT=NO,                                           X
               DBCTLID=IMSA,                                        X
               DDNAME=CCTLDD,                                       X
               DSNAME=IMS.SDFSRESL,                                 X
               MAXTHRD=99,                                          X
               MINTHRD=10,                                          X
               TIMER=60,                                            X
               USERID=,                                             X
               CNBA=10,                                             X
               FPBUF=,                                              X
               FPBOF=,                                              X
               TIMEOUT=60,                                          X
               SOD=A,                                               X
               AGN=
         END
/*
//LNKEDT EXEC PGM=IEWL,
//       PARM='LIST,XREF,LET,NCAL'
//SYSUT1 DD UNIT=SYSDA,SPACE=(1024,(100,50))
//SYSPRINT DD SYSOUT=*
//SYSLMOD  DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSLIN   DD DISP=(OLD,DELETE),DSN=&&OBJMOD
//         DD DDNAME=SYSIN
//SYSIN    DD *
   NAME  DFSPZP00(R)
/*
```

Figure 8. Example JCL to generate a DRA startup table

Customizing DBCTL

Use this information about facilities to help you customize DBCTL.

Installing and generating DBCTL in the IMS Database Control Guide

- "DFHDBUEX"
- "Global user exits XDLIPRE and XDLIPOST"
- "Global user exits XRMIIN and XRMIOUT" on page 32

DFHDBUEX

DFHDBUEX is an IBM-supplied user-replaceable program that is invoked each time CICS connects to, and disconnects from, DBCTL.

You can use DFHDBUEX to enable, or disable, CICS-DBCTL transactions at DBCTL connection and disconnection time. The transactions are available to be run if that DBCTL is connected. Users who attempt to enter one of these transactions when DBCTL is not connected are notified immediately that the transaction is unavailable. This means that end users will not be able to start one of these transactions, only to find that it fails because the database is unavailable.

To summarize, DFHDBUEX is invoked when:

- CICS has successfully connected to DBCTL.
- CICS is disconnecting from DBCTL, and has been notified that:
  - DBCTL has been terminated normally (using a /CHECKPOINT FREEZE or /CHECKPOINT PURGE command, as described in "Stopping DBCTL normally" on page 62).
  - The DRA has terminated abnormally.
  - DBCTL has terminated abnormally.
  - The menu transaction CDBC has been used to request disconnection from DBCTL.

See *CICS Customization Guide* for programming information on DFHDBUEX.

Global user exits XDLIPRE and XDLIPOST

The two global user exits XDLIPRE and XDLIPOST are available to all DL/I users, both remote users and DBCTL users. Use these global user exits to intercept any Call level or EXEC level DL/I request on entry to and exit from DL/I.

XDLIPRE is invoked before the DL/I request is processed. XDLIPOST is invoked after the DL/I request is processed. If you are using function shipping, the exits are invoked from the application owning region (AOR), and the database owning region (DOR). However, there are restrictions on what actions can be performed by an exit program running at exit point XDLIPRE or XDLIPOST in a DOR. For programming information about these exits, see "Naming, testing, and debugging your autoinstall control program" and "CICS action on return from the control program" in the *CICS Customization Guide*.

Programs running in these exits must be coded to threadsafe standards and defined to CICS as threadsafe.

Uses of the XDLIPRE and XDLIPOST global user exits

Use XDLIPRE to change the PSB name that the application program has scheduled at execution time. An example of XDLIPRE that you can modify is shown in “Using global user exit XDLIPRE to change PSB to be scheduled” on page 107.

Use the XDLIPRE exit to change the identity of the SYSID during CICS execution. You might want to change the identity of the SYSID if the one you are currently using becomes unavailable.

Use the XDLIPOST exit with DBCTL to ensure that all the required resources are available before an application starts. The enhanced scheduling feature in DBCTL allows a PSB to be scheduled when one or more databases are unavailable, you can use XDLIPOST to prevent this from happening. Use XDLIPOST to scan the list of PCBs and update the status of any unavailable databases to a response code of 0805. Setting the status of unavailable databases to 0805 means that CALLDLI programs return a value of 0805, EXEC DLI programs abend with code DHTE, and DBCTL does not raise any new schedule requests before the PSB is stopped.

Global user exits XRMIIN and XRMIOUT

The global user exits XRMIIN and XRMIOUT enable you to monitor activity across the resource manager interface (RMI).

XRMIIN is invoked just before control is passed from the RMI to a task-related user exit, and XRMIOUT is invoked just after control is returned to the RMI. You can use these exits to monitor DL/I activity; for example, control being passed to and from DFHDBAT for DBCTL requests, or DFHEDP for EXEC DLI.

For programming information on using these exits, see "Naming, testing, and debugging your autoinstall control program" and "CICS action on return from the control program" in the *CICS Customization Guide*.

Illustration of DBCTL startup parameter creation and selection

This illustration shows you how the DBCTL startup parameters are created and selected during startup.

If you are new to IMS system definition, use this figure while reading “Generating DBCTL” on page 17.

Note: “OCU” in Figure 9 on page 33 refers to the IMS online change utility.

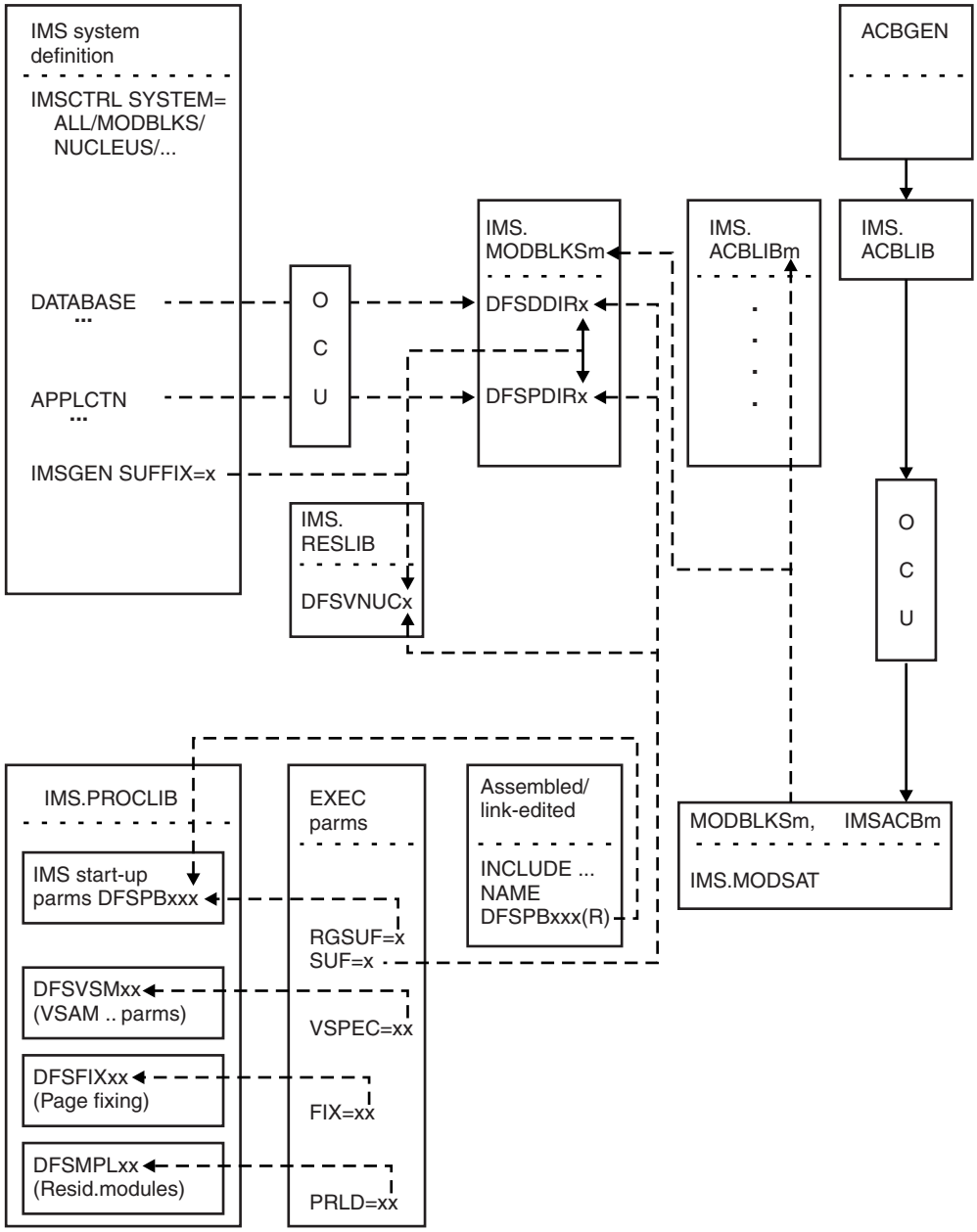


Figure 9. Creating and selecting DBCTL startup parameters

Chapter 3. Operations with DBCTL

This information shows you how to connect to DBCTL and issue operator commands.

Connecting to DBCTL: overview

You can perform CICS and DBCTL startup from a TSO terminal or an MVS console.

About this task

Before DBCTL can begin accepting transactions, several things must happen, as shown in Figure 10. The numbers in the figure and corresponding step numbers indicate the sequence of events.

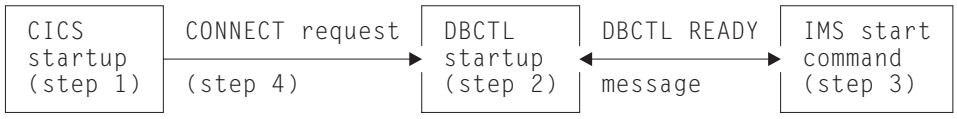


Figure 10. Connecting to DBCTL

1. CICS is started by submitting a job or starting a procedure, as described in *CICS Operations and Utilities Guide*.
2. DBCTL is started by submitting a job or starting a procedure, as described in "Starting DBCTL, DLISAS, and DBRC" on page 26.
3. After receiving a DBCTL READY message, indicating that startup is complete, the IMS console operator enters a start command, as follows:
   - If starting DBCTL for the first time, use /NRESTART CHECKPOINT 0 FORMAT ALL. This command cold starts DBCTL and formats the write ahead data set (WADS) and the restart data set (RDS).
   - /NRESTART for a warm start.
   - /ERESTART for an emergency restart after a failure.

   The / used in these commands is explained in "Operator communication with DBCTL: overview" on page 43. See "Restarting DBCTL" on page 66 for information about restart options.

   When the start has completed, the following message is issued:

   ```
   DFS994I rtype START COMPLETED
   ```

   where *rtype* is the type of start requested (COLD, WARM, or EMERGENCY).
4. The CICS operator requests connection to DBCTL using the CDBC transaction.

Step 1 can be done before, during, or after steps 2 and 3. Steps 2 and 3 must be done in the sequence described, and all three steps must be completed successfully before step 4 can begin.

The previous steps show you how to manually start IMS. IMS can also be started and restarted automatically, for more details see *IMS System Administration Guide* or *IMS Administration Guide: System*.

Connecting DBCTL to CICS automatically

You can specify that CICS is connected automatically to either the same or a different DBCTL.

If you want to connect automatically to the DBCTL that was being used when CICS was last shutdown, use the **DBCTLCON** system initialization parameter, or add an entry for DFHDBCON to the PLTPI so that it is invoked in the second stage of PLTPI processing (that is, the third stage of CICS initialization).

If you want to connect automatically to a specific DBCTL, or to connect CICS to DBCTL when it was not connected at shutdown, use the CICS **INITPARM** system initialization parameter, in addition to specifying DFHDBCON in the PLTPI. **INITPARM** enables DFHDBCON to have access to the DRA startup parameter table suffix you want to use. Specify:

```
INITPARM=(DFHDBCON='xx[,yyyy]')
```

where xx is a 1-to 2-character DRA startup table suffix, which you must enter, and yyyy is an optional 1-to 4-character DBCTL identifier. The DBCTL identifier specified in INITPARM overrides the DRA startup parameter DBCTLID.

Using **INITPARM** avoids the need to use the CRLP or DASD sequential terminal as your means of automating connection to a specific DBCTL. Use the following code if you prefer to use a CRLP or DASD sequential terminal:

```
//DDIN DD *
  CDBC CONNECT SUFFIX(xx) DBCTLID(yyyy)\
```

where xx is the 1- to 2-character DRA startup table suffix and yyyy is the 1- to 4-character DBCTL identifier, both of which are optional. Specifying a DBCTL identifier here overrides the one specified in the DRA startup table parameter DBCTLID. \ is the end-of-line character. (See "DFHLIST definitions" in the *CICS Resource Definition Guide* and "Using sequential terminal support" in the *CICS Application Programming Guide* for guidance on using sequential terminal support.)

What happens at startup depends on the type of CICS start being used, whether you specified **INITPARM**, and whether DBCTL was connected to CICS when CICS was last shutdown.

Connecting to DBCTL after a CICS WARM or EMERGENCY start

If CICS startup is WARM or EMERGENCY:

- If you used INITPARM, the DRA startup table suffix and DBCTL identifier specified there are used to determine which DBCTL to connect to, whether CICS and DBCTL were connected when CICS was last shutdown.
- If you did not use INITPARM:
  - If CICS and DBCTL were connected when CICS was last shutdown, CICS is reconnected to the same DBCTL. DFHDBCON uses the DRA startup parameter table suffix and DBCTL identifier override (which might be blanks) from the catalog.
  - If CICS and DBCTL were ***not*** connected when CICS was last shutdown CICS issues message DFHDB8117 and does not attempt to connect to DBCTL.

Connecting to DBCTL after a CICS COLD or INITIAL start

If CICS startup is COLD or INITIAL:

- If you used INITPARM, CICS attempts to connect to DBCTL, using the suffix and DBCTL identifier (if any) you specified.
- If you did not use INITPARM, CICS attempts to connect to DBCTL using the default DRA startup table suffix (00) and no DBCTL identifier override, whether DBCTL was connected when CICS was last shutdown.

Connection, disconnection, and inquiry transactions for the CICS DBCTL interface

There are two CICS transactions that you can use to connect to, disconnect from, and inquire on the status of the CICS-DBCTL interface.

They are:

- CDBC, which enables users (for example, CICS operators and network controllers) to display a menu to connect to and disconnect from DBCTL.
  - For ***connection***, CDBC issues a DBCTL connection request to DFHDBAT, which issues a DRA INIT request internally to the DRA.

    CDBC also enables you to override the DRA startup parameter table suffix and DBCTL identifier when you are connecting CICS to DBCTL. (See "Defining the IMS DRA startup parameter table" on page 27 for information on the contents of the DRA startup table.)
  - For ***disconnection***, CDBC can issue an orderly or an immediate disconnection request to DFHDBAT, which issues a DRA TERM request internally to the DRA.

    (See "CDBC transaction for connect and disconnect" for more information on using CDBC.)
- CDBI, which enables users to inquire on the status of the CICS-DBCTL interface. See "CDBI transaction for inquiry" on page 41 for more information.

You can enter CDBC and CDBI from either a CICS terminal or an MVS console. You can restrict access to these transactions using transaction security. Messages from CDBC can be sent to the transient data destination CDBC.

CDBC transaction for connect and disconnect

Typing CDBC on a 3270-type terminal displays a menu for connecting CICS to, and disconnecting it from, DBCTL.

Figure 11 on page 38 shows an example of the menu.

To connect to DBCTL, enter option number 1 after:

```
Option Selection ==>
```

```
CDBC               CICS-DBCTL CONNECTION/DISCONNECTION                93.259
                                                                     13:39:20

          Select one of the following:

               1 Connection
               2 ORDERLY disconnection
               3 IMMEDIATE disconnection


          Option Selection     ==> 2
          Startup Table Suffix ==> 00
          DBCTL ID Override    ==>

  DFHDB8209D DBCTL orderly disconnection requested. Press PF5 to confirm.

  Status of the Interface: DFHDB8293I DBCTL connected and ready.
              CICS APPLID: IYAHZCD2
                 DBCTL ID: SYS2
     Startup Table Suffix: 00


PF1 = Help   2 = Refresh   3 = End
```

Figure 11. CDBC transaction menu screen

If you want to specify a DRA startup table suffix, you can enter it after:
`Startup Table Suffix ==>`

If you do not specify a suffix, CICS uses the one that was used when it was last connected to DBCTL. If this is the first time you have connected CICS to DBCTL, and you do not specify a suffix, CICS uses the default suffix, which is 00.

If you want to specify a DBCTL identifier, you can enter it after:
`DBCTL ID Override ==>`

If you do not specify a DBCTL identifier, the DRA uses the DBCTL identifier specified on the DBCTLID parameter in the DRA startup table.

When you have pressed ENTER, you should get the message:
`DFHDB8209 I DBCTL orderly disconnection requested. Press PF5 to confirm.`

as shown on the example screen in Figure 11.

The CDBC menu screen displays the following additional information:

- Status of the CICS-DBCTL interface; in this case, DBCTL is connected and ready
- The APPLID of the CICS system; in this case, DBDCCICS
- The identifier of the DBCTL system; in this case, SYS2
- The DRA startup parameter table suffix for this connection; in this case, 00.

The DBCTL identifier and the DRA startup parameter table suffix are only displayed when CICS has been connected to DBCTL. You can refresh any of the information on the CDBC menu screen by pressing PF2.

You can obtain a help screen for the CDBC menu by pressing PF1. As you can see in Figure 12 on page 39, the CDBC help screen reminds you which number to specify for which option, what the options mean, and summarizes the CICS-DBCTL interface information displayed on the CDBC menu screen.

```
               HELP : CICS-DBCTL CONNECTION/DISCONNECTION

To CONNECT to DBCTL, select option 1. You can also specify a startup
 table suffix, or accept the existing suffix. The id of the DBCTL system is
 obtained from the startup table, but can be optionally overridden.

To DISCONNECT from DBCTL, select option 2 or option 3.

   Select option 2 for ORDERLY disconnection: this allows all CICS-DBCTL
   transactions from this CICS to complete before disconnecting from DBCTL.

   Select option 3 for IMMEDIATE disconnection: this allows all CICS-DBCTL
   requests from this CICS to complete before disconnecting from DBCTL.
--------------------------------------------------------------------------------
Displayed information (press PF2 to refresh the information):
   STATUS OF THE INTERFACE The current status of the connection to DBCTL.
   CICS APPLID             The application identifier for this CICS system.

   Displayed when available:
   DBCTL ID                 Identifier of the DBCTL system with which this
                            CICS system is communicating.
   STARTUP TABLE SUFFIX     Suffix used when CICS was connected to DBCTL.

                 PRESS ENTER TO RETURN TO SELECTION SCREEN
```

Figure 12. CDBC transaction menu help screen

Using CDBC without the menu screen

About this task

The menu screen is displayed if you use CDBC from a 3270-type terminal, However, if you issue CDBC from a CRLP or DASD sequential terminal or operating system console, the menu screen is **not** displayed. For example, if you specify:

```
CDBC CONnect
```

DBCTL is connected using the default suffix, 00.

If you specify a suffix:

```
CDBC CONnect SUFfix(12)
```

and DBCTL is connected using suffix 12.

You can also type a DBCTL identifier, in addition to the suffix, or on its own. For example, if you enter:

```
CDBC CONnect DBCtlid(DBC1)
```

CICS is connected to the DBCTL named DBC1.

You can also enter:

```
CDBC CONnect DBCtlid(DBC2) SUFfix(11)
```

or

```
CDBC CONnect SUFfix(11) DBCtlid(DBC2)
```

in either case, CICS is connected to DBCTL DBC2, using suffix 11.

See “What happens when you have requested connection to DBCTL” on page 40 below for details of the system’s response to your connection request.

If you disconnect CICS from DBCTL using a BSAM CRLP-type terminal, the menu screen is not displayed.

For orderly disconnection, specify:

```
CDBC DISconnect
```

For immediate disconnection, enter:

```
CDBC DISconnect IMMediate
```

See "Deciding whether to use orderly or immediate disconnection" on page 41 for information on the two types of disconnection request.

What happens when you have requested connection to DBCTL

When you have requested connection to DBCTL, you get messages confirming that connection is taking place.

If you have used the CDBC menu, the following messages are displayed on the terminal:

```
Status of the Interface: DFHDB8292I DBCTL CONNECT PHASE 2 IN PROGRESS.
Status of the Interface: DFHDB8293I DBCTL CONNECTED AND READY.
```

If you have not used the CDBC menu, the following messages are displayed on the MVS console:

```
+DFHDB8210D CONNECTION TO DBCTL IS PROCEEDING. CHECK CDBC TD QUEUE.
+DFHDB8225I DBDCCICS THE DBCTL ID IS SYS1. THE DRA STARTUP TABLE SUFFIX IS 00.
```

CICS-DBCTLDFHDBnnnn messages that are issued when you are using CDBC.

If DBCTL is not yet available, the main CICS-supplied IMS control exit, DFHDBCTX, is invoked. DFHDBCTX in turn calls DFHDXAX. For more information about the IMS control exit routines, see the appropriate *IMS Customization Guide*.

For a DBCTL restart, the control exit is invoked as for any DBCTL connection attempt. However, instead of returning control directly to the DRA, the control transaction invokes the DFHDXAX module. This control exit routine checks to see if it is being invoked for a failing connection:

- If it is not being invoked for a failing connection, it does not attempt to connect and passes back control.
- If it is being invoked for a failing connection, it checks the input arguments to determine whether:
  - An IDENTIFY attempt failed, and
  - CICS is not in the process of terminating

If an IDENTIFY failed, and CICS is not terminating, DFHDXAX selects the current DBCTL ID, and initiates repeated attempts to reconnect to the current DBCTL, thus avoiding operator intervention.

Retries are made every five seconds for a ten minute period, and message DFHDB8297 is issued periodically. If reconnection is still not successful after ten minutes, DFHDXAX abandons the attempt, and requests IMS to issue message DFS0690A, which requires operator intervention. The *IMS Messages and Codes* contains guidance on interpreting the messages that are displayed when you are

using CDBC. If you reply CANCEL, the connection attempt is abandoned. It you reply WAIT, the DRA attempts to connect again after the number of seconds specified in the **TIMER** parameter in the DRA startup parameter table. If the connection attempt fails again, the DRA continues to attempt to connect after the same number of seconds. You can stop these repeated connection attempts by using the CDBC transaction to disconnect from DBCTL. You can use either the same instance of CDBC or run the transaction on a different terminal. Disconnection takes effect when the DRA next tries to reconnect to DBCTL.

Deciding whether to use orderly or immediate disconnection

Use immediate disconnection only if necessary. For example, you may need to use it if you have already issued an orderly disconnection request which has not taken place, and you need disconnection to take place soon.

Orderly disconnection allows all existing CICS-DBCTL tasks to complete before CICS is disconnected from DBCTL. Tasks not currently using DBCTL are prevented from issuing further PSB schedule requests. This means that there should not be any indoubt logical units of work (UOWs), and database records are available to other CICS systems connected to that DBCTL.

Immediate disconnection allows only current DL/I requests to DBCTL from this CICS system to complete before CICS is disconnected from DBCTL. Any new DL/I or PSB schedule requests are prevented. This can cause indoubt UOWs for the task involved and leave database records unavailable for other CICS systems connected to that DBCTL until it is reconnected. What happens depends on the type of request issued to DBCTL after the immediate disconnection request:

- If it is a PSB schedule request, a DHTJ abend (for a command-level program) or a DLINA condition (for a call-level program) is issued.
- If it is a DL/I request, the UOW is backed out and an ADCA abend is issued.
- If it is a PREPARE request, the UOW is backed out and an ASP7 abend is issued.

  In all the above cases, database records are available to other applications.
- If it is a COMMIT request, the task remains indoubt and DBCTL records are unavailable. The in-doubts will not be resolved until DBCTL is reconnected to CICS. An abend is issued when the next PSB schedule is received, as described for PSB schedule request, above.

  See “Two-phase commit for DBCTL” on page 70 for information on PREPARE and COMMIT requests.

So, use immediate disconnection only if necessary. For example, you may need to use it if you have already issued an orderly disconnection request which has not taken place, and you need disconnection to take place soon. Orderly disconnection may be delayed by a task that is issuing many DL/I requests, or by a conversational task that is awaiting input from an unattended terminal. If you think the problem is being caused by such a task, you may prefer to identify it using CEMT INQ TASK, and then use CEMT SET TASK(n) PURGE, where “n” is the task identifier to purge it. You can then use orderly disconnection. However, if the problem is being caused by many tasks or by a single task that you cannot identify, you may have to use immediate disconnection.

CDBI transaction for inquiry

You can use the CDBI transaction to inquire on the status of the DBCTL connection.

Typing CDBI displays a screen like the one shown in Figure 13. The CDBI screen shows the status of the CICS-DBCTL interface (in this example, DBCTL is connected and ready), plus the APPLID of the CICS system (DBDCCICS) and the DBCTL identifier (SYS1). You can refresh the information by pressing PF2.

```
CDBI                    CICS-DBCTL INTERFACE INQUIRY            93.194
                                                               11:23:50
            Status     : DFHDB8293 I DBCTL connected and ready.
            CICS APPLID: DBDCCICS
            DBCTL ID   : SYS1









PF1 = Help   2 = Refresh   3 = End
```

Figure 13. CDBI transaction screen

You can obtain a help screen for CDBI by pressing PF1. Figure 14 shows an example of such a panel. The CDBI help screen tells you how to refresh the information on the CDBI screen, and explains that information. It includes a list of the CICS messages describing the status of the CICS-DBCTL interface that can appear on the CDBI screen.

```
                    HELP : CICS-DBCTL INTERFACE INQUIRY
 The CICS-DBCTL interface inquiry screen shows:
   STATUS OF THE INTERFACE    The status can be:
     DFHDB8290I DBCTL NOT CONNECTED TO CICS.
     DFHDB8291I DBCTL CONNECT PHASE 1 IN PROGRESS.
     DFHDB8292I DBCTL CONNECT PHASE 2 IN PROGRESS.
     DFHDB8293I DBCTL CONNECTED AND READY.
     DFHDB8294I DBCTL ORDERLY DISCONNECT IN PROGRESS.
     DFHDB8295I DBCTL IMMEDIATE DISCONNECT IN PROGRESS.
     DFHDB8296I DBCTL CANNOT BE CONNECTED TO CICS.
   CICS APPLID  The application identifier of this CICS system.
 Displayed when available:
   DBCTL ID     The identifier of the DBCTL system with which this CICS
                is communicating
 You can press PF2 to update (refresh) the information shown on the screen









                 PRESS ENTER TO RETURN TO INQUIRY SCREEN
```

Figure 14. CDBI transaction help screen

Operator communication with DBCTL: overview

IMS operations can be done from an IMS master terminal operator console, which is usually the primary MVS console.

This can be the primary MVS console, but you are advised to have a secondary MVS console that is dedicated to DBCTL, this dedicated console is called the DBCTL console.

You can choose to issue operator commands to DBCTL from a CICS terminal, using a CICS-supplied transaction, CDBM, as described in "CDBM operator transaction" on page 47.

Use the Resource Access Control Facility (RACF) to control access to IMS resources. For further information about using RACF see *IMS System Administration Guide* or *IMS Administration Guide: System*.

This section covers:

DBCTL operator commands

The operator commands you can use to communicate with DBCTL are a subset of IMS operator commands.

This book summarizes the ways in which you can use these commands with DBCTL. For guidance on syntax, see *IMS Operator's Reference* . See also "Summary of DBCTL operator commands" on page 45 for a list of DBCTL operator commands and their corresponding CICS commands, and a list of valid keywords for DBCTL users.

Format of DBCTL operator commands

The format of DBCTL operator commands is to being with a command recognition character (CRC), followed by a verb, a password (if required), a keyword or keywords, and optional comments.

DBCTL commands begin with a command recognition character (CRC). A CRC of / is the default. (The examples of DBCTL commands in this manual use the default CRC.) You can override it on the DBCTL job, but remember that ***each DBCTL subsystem in a single MVS image must have a different CRC***. This CRC must also be different from every other subsystem in the processor (or multiprocessor), not just DBCTL subsystems. The same applies to any test systems you might be using. You can, if you prefer, use the subsystem ID (for example, SYS1) of the DBCTL you are using instead of a CRC.

There must be no space between the CRC and the verb. Usually there is a space between parameters, except as noted for specific parameters in *IMS Operator's Reference*. Many verbs and keywords have abbreviations. Guidance on using them is in *IMS Operator's Reference*.

Multisegment DBCTL operator commands

The DBCTL operator commands /CHANGE, /ERESTART, /RMxxxxxx, and /SSR can be entered in multiple segments.

The format of multisegment commands varies according to the environment you are using. For multisegment commands in a DBCTL environment, each segment preceding the last segment requires an end-of-**segment** (EOS) indicator, which is the CRC followed by the ENTER key. The last (or only) segment requires an end-of-**message** (EOM) indicator, which is the ENTER key. In addition, each segment must begin with the CRC.

Figure 15 on page 45 is an example of a multisegment command that has two segments. The CRC is a slash (/), and appears at the beginning and end of the first segment. The EOS of the first segment is the CRC (/) followed by the ENTER key, which does not appear because it is not displayable. The EOM of the second (and last) segment is the ENTER key, so this segment begins with the CRC, but does not end with it.

DBCTL can handle single-segment commands from an unlimited number of consoles concurrently, but the number of consoles that can concurrently issue multisegment commands is limited to eight. A single multisegment command is limited to 241 bytes. If either of these limits is exceeded, a message is sent to the issuing console.

```
/RMI DBRC='ic dbd(dedbdd01) area(dd01ar0) icdsn(fvt31.dedbdd01.dd01ar0
.ic.dummy1) icdsn2/
/(FVT31.DEDBDD01.DD01AR0.IC2.DUMMY1) HSSP'
DFS000I MESSAGE(S) FROM ID=SYS1 490
INIT.IC DBD(DEDBDD01) AREA(DD01AR0) -
ICDSN(FVT31.DEDBDD01.DD01AR0.IC.DUMMY1) -
ICDSN2(FVT31.DEDBDD01.DD01AR0.IC2.DUMMY1) HSSP
DSP0203I  COMMAND COMPLETED WITH CONDITION CODE 00
DSP0220I  COMMAND COMPLETION TIME 89.045   16:24:58.7
DSP0211I  COMMAND PROCESSING COMPLETE
DSP0211I  HIGHEST CONDITION CODE = 00
DSP0058I  RMI COMMAND COMPLETED
/RMI DBRC='ic dbd(dedbdd01) area(dd01ar0) icdsn(fvt31.dedbdd01.dd01ar0
.ic.dummy2) /
/ICDSN2(FVT31.DEDBDD01.DD01AR0.IC2.DUMMY2) HSSP'
DFS000I MESSAGE(S) FROM ID=SYS1 514
INIT.IC DBD(DEDBDD01) AREA(DD01AR0) -
ICDSN(FVT31.DEDBDD01.DD01AR0.IC.DUMMY2) -
ICDSN2(FVT31.DEDBDD01.DD01AR0.IC2.DUMMY2) HSSP
DSP0203I  COMMAND COMPLETED WITH CONDITION CODE 00
DSP0220I  COMMAND COMPLETION TIME 89.045   16:28:10.3
DSP0211I  COMMAND PROCESSING COMPLETE
DSP0211I  HIGHEST CONDITION CODE = 00
DSP0058I  RMI COMMAND COMPLETED
```

Figure 15. Example of using multisegment commands in a DBCTL environment

For further guidance on multisegment operator commands, see *IMS Operator's Reference*.

You can use null words (for example, FOR, and TO) within the operator commands to help clarify the syntax without affecting the command itself. Because null words are reserved, you must not use them to name system resources. For further guidance on null words, see *IMS Operator's Reference*.

You might need to use a password depending on the security facility used. See Chapter 6, "Security checking with DBCTL," on page 117 for information about security considerations with DBCTL.

Summary of DBCTL operator commands

The following tables show you the CICS operator commands, corresponding DBCTL operator commands, and which DBCTL commands can be issued using the CICS-supplied transaction CDBM. Also shown are the IMS operator commands and keywords valid with DBCTL.

Chapter 3, "Operations with DBCTL," on page 35 and Chapter 4, "Recovery and restart operations for DBCTL," on page 65 contain information about using operator commands with DBCTL. For further guidance on the syntax of DBCTL operator commands, see *IMS Operator's Reference*.

Note: The / used in these commands is the **default** command recognition character (CRC). For information about the usage of CRCs, see "Operator communication with DBCTL: overview" on page 43.

Table 2. DBCTL operator commands and CICS equivalents

DBCTL operator command	CICS equivalent	Valid with CDBM
/CHANGE	None	Yes

Table 2. DBCTL operator commands and CICS equivalents (continued)

DBCTL operator command	CICS equivalent	Valid with CDBM
/CHECKPOINT (simple form)	ACTIVITY KEYPOINT	Yes
/CHECKPOINT FREEZE or /CHECKPOINT PURGE	CEMT PERFORM SHUTDOWN	No
/CHECKPOINT STATISTICS	CEMT PERFORM STATISTICS RECORD	Yes
/DBDUMP	None	Yes
/DBRECOVERY	None	Yes
/DELETE	None	Yes
/DEQUEUE	None	Yes
/DISPLAY ACTIVE or /DISPLAY CCTL	CEMT INQUIRE TASK	Yes
/DISPLAY DATABASE	None	Yes
/DISPLAY DBD, /DISPLAY POOL, and /DISPLAY PSB	None	Yes
/ERESTART	SIT with START=AUTO resulting in EMER restart	No
/LOCK	None	Yes
/LOG	None	Yes
/MODIFY	None	No
/NRESTART CHECKPOINT 0	SIT START=INITIAL	No
/NRESTART (without CHECKPOINT 0)	SIT with START=AUTO resulting in WARM start	No
/PSTOP	None	Yes
/RMCHANGE	None	Yes
/RMDELETE	None	Yes
/RMGENJCL	None	Yes
/RMINIT	None	Yes
/RMLIST	None	Yes
/RMNOTIFY	None	Yes
/SSR	None	No
/START DATABASE	None	Yes
/STOP DATABASE	None	Yes
/STOP THREAD	CEMT SET TASK PURGE	Yes
/SWITCH OLDS	None	Yes
/TRACE SET PI	None	Yes
/UNLOCK	None	Yes
/VUNLOAD	None	Yes
MVS MODIFY jobname,RECONNECT	CEMT PERFORM RECONNECT	N/A: MVS command
MVS MODIFY jobname,STOP|DUMP	CEMT PERFORM SHUTDOWN IMMEDIATE	N/A: MVS command

Table 3. DBCTL operator commands and keywords

DBCTL operator command	Keyword(s)
/CHANGE	CCTL, PASSWORD, SUBSYS
/CHECKPOINT	FREEZE, PURGE, ABDUMP, SNAPQ
/DBDUMP	DATABASE
/DBRECOVERY	AREA, DATABASE
/DELETE	DATABASE, PASSWORD, PROGRAM
/DISPLAY	ACTIVE, AREA, CCTL, DATABASE, DBD, INDOUBT, MODIFY, OASN SUBSYS, OLDS, POOL, PROGRAM, PSB, SHUTDOWN STATUS, STATUS, TRACE
/ERESTART	CHECKPOINT, COLDBASE, COLDCOMM, COLDSYS, FORMAT, NOBMP
/LOCK	DATABASE, PROGRAM
/LOG	None
/MODIFY	ABORT, COMMIT, PREPARE
/NRESTART	CHECKPOINT 0, FORMAT, NOPASSWORD, PASSWORD
/PSTOP	REGION
/RMCHANGE	DBRC modifier
/RMDELETE	DBRC modifier
/RMGENJCL	DBRC modifier
/RMINIT	DBRC modifier
/RMLIST	DBRC modifier
/RMNOTIFY	DBRC modifier
/SSR	Commands and keywords from appropriate subsystem (for example, DB2)
/START	AREA, AUTOARCH, DATABASE, OLDS, PROGRAM, REGION | THREAD[1], WADS
/STOP	ADS, AREA, AUTOARCH, DATABASE, OLDS, PROGRAM, REGION | THREAD[1], WADS
/SWITCH	OLDS
/TRACE	SET, MONITOR, PI, PSB, TABLE
/UNLOCK	DATABASE, PROGRAM
/VUNLOAD	AREA

Note: THREAD is a synonym for REGION.

CDBM operator transaction

You can use CDBM to issue most of the IMS operator commands that are valid for DBCTL across the DRA interface to DBCTL to display and change the state of selected resources.

CDBM also provides a means of maintaining a command file which stores commands. You can store commands for any reason, most likely because you want to reuse them. These stored commands can include more databases than the operator transaction panel has space for.

When dealing with databases, you can use an asterisk (*) to refer to generic groups; for example DB21* refers to all databases starting with the characters

DB21. You can also use a plus (+) sign in place of a single character; for example, DB+2 displays databases DB12, DB22, DB32, and so on.

You can issue DBCTL commands via a menu panel, as shown in Figure 16. This panel is obtained by starting the CDBM transaction.

```
CDBM                CICS-DBCTL Operator Transaction              98.135
                                                                13:24:20

Type IMS command.
            ____________________________________________________________
            ____________________________________________________________
            ____________________________________________________________
            ____________________________________________________________


For /DBDUMP or /DBRECOVER commands

Choose one. 1 1. Do not force end of volume
              2. Force end of volume


Press enter to display responses.


CICS APPLID DBDCCICS
DBCTL ID    SYS3


F1=Help  F2=Maintenance  F3=Exit  F5=Refresh  F12=Cancel
```

Figure 16. CDBM CICS-DBCTL operator transaction panel

On this panel you can enter a DBCTL command, for example:

```
/DISPLAY DB ALL
```

or a group command, for example:

```
/GROUP SAMPLE STA
```

There is also a help screen, as shown in Figure 17 on page 49.

```
 CDBM               Help: CICS-DBCTL Operator Transaction


 CDBM           Use the transaction to send an IMS command to a DBCTL system.

 Command        Type the command recognition character / followed by an IMS
                command and press enter to display responses.

 Responses      Use the PF keys to page IMS responses.

 Wildcards      * or + can be used within one database name.

 End of volume  For /DBDUMP or /DBRECOVER commands only
                Choose one.
                  1. Do not force end of volume
                  2. Force end of volume
  CICS APPLID
               These are shown for information.
 DBCTL ID
 Enter the group common maintenance screen.


 Example        /DIS DB DEPT* displays the status of several databases.


 F3=Exit  F12=Cancel
```

Figure 17. CDBM CICS-DBCTL operator transaction help panel

An example of the use of a /GROUP command from the CICS-DBCTL Operator Transaction screen is shown in Figure 18.

```
 CDBM                  CICS-DBCTL Operator Transaction              98.135
                                                                   13:24:20
 Type IMS command.
             /GROUP SAMPLE STA_____________________________________________
                ______________________________________________________________
                ______________________________________________________________
                ______________________________________________________________


For /DBDUMP or /DBRECOVER commands

Choose one. 1 1. Do not force end of volume
              2. Force end of volume


 Press enter to display responses.


 CICS APPLID DBDCCICS
 DBCTL ID    SYS3


F1=Help  F2=Maintenance  F3=Exit  F5=Refresh  F12=Cancel
```

Figure 18. CICS-DBCTL operator transaction panel showing a GROUP command

Responses to commands issued from the CDBM screen are returned on a screen like the one in Figure 19 on page 50, which shows the first of a number of screens resulting from a /DISPLAY DB ALL command.

```
CDBM                    CICS-DBCTL IMS Responses                Screen 1
                                                      Responses 1    to 18
                                                          More:   +
DATABASE  TYPE  TOTAL UNUSED  TOTAL UNUSED ACC  CONDITIONS
ACCOUNDB                                   UP   STOPPED, NOTOPEN, NOTINIT
ADMIDX1                                    UP   STOPPED, NOTOPEN, NOTINIT
ADMOBJ1                                    UP   STOPPED, NOTOPEN, NOTINIT
ADMOBJ2                                    UP   STOPPED, NOTOPEN, NOTINIT
ADMOBJ3                                    UP   STOPPED, NOTOPEN, NOTINIT
ADMSYSDF                                   UP   STOPPED, NOTOPEN, NOTINIT
BE1CHKPT  DL/I                             UP   NOTOPEN
BE1PARTA                                   UP   STOPPED, NOTOPEN, NOTINIT
BE1PARTB                                   UP   STOPPED, NOTOPEN, NOTINIT
BE1PARTC                                   UP   STOPPED, NOTOPEN, NOTINIT
BE1PARTS                                   UP   STOPPED, NOTOPEN, NOTINIT
BE2ORDER  DL/I                             UP   NOTOPEN
BE2ORDRX  DL/I                             UP   NOTOPEN
BE2PARTS  DL/I                             UP   NOTOPEN
BE2PCUST  DL/I                             UP   NOTOPEN
BE3ORDER  DL/I                             UP   NOTOPEN
BE3ORDRX  DL/I                             UP   NOTOPEN
                                                          More...

F1=Help  F3=Exit  F4=Top  F6=Bottom  F7=Bkwd  F8=Fwd  F9=Retrieve  F12=Cancel
```

Figure 19. CDBM CICS-DBCTL IMS responses panel

Alternatively, you can issue CDBM and the DBCTL command directly, as follows:

```
CDBM /xxxxxxxx
```

where / is the default CRC and xxxxxxxx is an IMS operator command that is valid for use with DBCTL and CDBM.

Note: IMS requires that each command is prefixed with the default CRC. The CRC is present only for syntax checking; it does not determine to which DBCTL the command is sent. You cannot use a CRC value to route a command to a particular DBCTL system through CDBM. It can be sent only to the one currently connected to CICS. This DBCTL can have its own CRC value which is different from the default one of '/'. However, this does not matter to CDBM, because the '/' character is used only for syntax checking, and the command is presented to the connected DBCTL without a CRC, using the AIB interface.

The /GROUP can also be entered in this way, for example:

```
CDBM /GROUP  SAMPLE  DIS.
```

The following IMS operator commands are valid with CDBM:

- /CHANGE
- /CHECKPOINT (simple form) and /CHECKPOINT STATISTICS
- /DBDUMP
- /DBRECOVERY
- /DELETE
- /DEQUEUE
- /DISPLAY
- /LOCK
- /LOG
- /PSTOP
- /RMCHANGE
- /RMDELETE

- /RMGENJCL
- /RMINIT
- /RMLIST
- /RMNOTIFY
- /START
- /STOP
- /SWITCH OLDS
- /TRACE SET PI
- /UNLOCK
- /VUNLOAD

The following IMS operator commands are not valid with CDBM and must be issued via the MVS console:

- /CHECKPOINT FREEZE and /CHECKPOINT PURGE
- /MODIFY
- /ERESTART
- /NRESTART
- /SSR

DFHDBFK - The CDBM GROUP command file

Before you can use the /GROUP command CDBM requires a file in which all your predefined commands can be stored. This file, DFHDBFK, is the CDBM GROUP command file. It is a VSAM KSDS.

Note: The DFHDBFK file must be defined as a local file to each region that uses the CDBM transaction. It cannot be shared by multiple regions. If the file is remote, the CDBM transaction receives an error when it attempts to open the file.

The DFHDBFK file is not required until you first attempt to use the /GROUP command.

Table 4. Record layout in the CDBM GROUP command file

Field	Length	Content	Description
1	12	Group	A 12-character field containing your chosen name for this group. The acceptable characters are A-Z 0-9 $ @ and #. Leading or embedded blanks are not allowed, but trailing blanks are acceptable.
2	10	IMS Command	A 10-character field containing any of the IMS command verbs that are valid for CDBM (see Commands valid with CDBM for details). Leading or embedded blanks are not allowed, but trailing blanks are acceptable. **Note:** The validity of the IMS command verb is not checked by CDBM. Invalid values will be reported by IMS when the command is attempted.

Table 4. Record layout in the CDBM GROUP command file (continued)

Field	Length	Content	Description
3	1406	IMS Command parameters	Up to 1406 characters of parameters appropriate to the chosen IMS command verb. (This will often consist of lists of databases.) **Note:** Wildcard characters may not be used in the parameters stored in the CDBM Group command file. This is unlike the other functions of the CDBM transaction which permit the use of wildcard characters to describe multiple similarly named databases.

Record layout in the CDBM GROUP command file

Each record in the DFHDBFK file contains one field which can be up to 1428 characters long.

The MAINTENANCE panel for DFHDBFK

If you press the Maintenance key (PF2) on the main CDBM panel, you get the panel shown in Figure 20.

```
CDBM               CICS/DBCTL COMMAND GROUP MAINTENANCE
_        ACTION      A add  B browse  D delete  R read  U update
____________ GROUP          __________ IMS COMMAND
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
>                                                              <
F1=Help  F3=Exit  F12=Cancel
```

Figure 20. CICS-DBCTL Group Maintenance Panel

Input fields

The input fields are:

- Action
- Group
- IMS Command
- IMS Command parameters

  (between the > < marks).

Group, IMS Command and IMS Command parameters are described in "Record layout in the CDBM GROUP command file"

The Action field will accept one of the following:

A Add

Add a new record to the DFHDBFK file. If the key already exists, the Add fails.

Note: To Add a record that is very similar to an existing record, but which has a different key, you may find it helpful to Read the existing record, modify the displayed fields, and then Add this new record.

B Browse

Displays the contents of the command file, record by record. Specify any key (or none) to indicate where you want the browse to start. Each time you press ENTER, Browse moves on to the next record. At the end of the file you will be prompted to wrap around to the start of the file. You can accept this or not as you prefer. Incomplete keys, and unknown keys are also acceptable as start points. If no key is provided, the browse starts at the first record in the file.

If you have used Browse to locate a specific record for deletion or for update, remember to use Read before either Delete or Update.

D Delete

Delete a record from the DFHDBFK file. A Delete must be immediately preceded by a Read to lock the required record.

R Read

Read displays a specific record. Unlike Browse it does not operate on partial, or absent keys, and does not present the next record when you press ENTER.

Read is required before those actions (Delete and Update) which change an existing record. It locks that record against the possibility of being changed by another operator. This action also serves to help you confirm that the correct record has been selected.

A lock is released by ending CDBM, or by your next CDBM Maintenance action (whether that is the Update or Delete you had contemplated, or something different entirely).

U Update

Update a record in the DFHDBFK file. An Update must be immediately preceded by a Read to lock the required record.

You cannot update the key fields (GROUP and IMS COMMAND).

Reminder:: Use Add to create a new key.

Note: In the descriptions above, *Key* refers to the 22 characters at the beginning of each record in the DFHDBFK file (namely the GROUP and IMS COMMAND).

If you press the help key (PF1) from the CICS-DBCTL Maintenance panel, you get the panel shown in Figure 21 on page 54.

```
 CDBM               Help: CICS-DBCTL Operator Transaction


Maintenance    Store commands for issuing from the CDBM screen.

GROUP          Enter the group you want to store a command in.

IMS COMMAND    Enter a valid IMS command to execute with the supplied data

ACTION         A - Add a command to the command file.
               B - Browse the contents of the command file.
               D - Delete a command, only after it has been read.
               R - Read a command from the file.
               U - Update a command, only after it has been read.

               Issue commands from the main screen in the format
               /GROUP group command.

Example        /GROUP SAMPLE DIS shows information for the databases in


F3=Exit  F12=Cancel
```

Figure 21. CICS-DBCTL Maintenance help panel

Issuing DBRC commands

With DBCTL, you must issue DBRC commands via DBCTL console commands (/RMxxxxxx) because DBRC runs outside the CICS address space. You can issue the /RMxxxxxx commands via the CICS-supplied transaction CDBM.

You can use the following /RMxxxxxx commands ***online***:

- /RMCHANGE: to change or modify information in the RECON
- /RMDELETE: to delete information from the RECON
- /RMGENJCL: to generate JCL for a specified utility
- /RMINIT: to create records in the RECON
- /RMLIST: to list the contents of the RECON
- /RMNOTIFY: to add information to the RECON.

For example:

```
/RMINIT DBRC='DB DBD(IVPDB2) SHARELVL(3)'.
```

See *IMS Operator's Reference* for further guidance on the syntax of these commands.

You can also enter DBRC commands in ***batch***, but the syntax is slightly different, as shown in Figure 22 on page 55.

```
//INITDB JOB 1,PGMERID,CLASS=Q,MSGCLASS=A
//*
//RECON    EXEC PGM=DSPURX00,REGION=1000K
//STEPLIB  DD DSN=IMS.RESLIB,DISP=SHR
//DFSRESLB DD DSN=IMS.RESLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//RECON1   DD DSN=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//SYSIN    DD *
 INIT.DB DBD(IVPDB2) SHARELVL(3)
/*
```

Figure 22. Example JCL to register a database with DBRC

IMS password security

Use the Resource Access Control Facility (RACF) to protect your databases and program specification blocks (PSBs).

RACF is part of the z/OS Security Server and can be used to control access to IMS resources. RACF has superseded the IMS Security Maintenance Utility (SMU) which was last supported in IMS version 9. For further information about password security see *IMS System Administration Guide* or *IMS Administration Guide: System*.

Controlling tracing of DBCTL events

To start and stop tracing of internal DBCTL events dynamically, and define activities to be monitored by the IMS monitor, use the /TRACE command.

About this task

- The PI keyword specifies that program isolation (PI) trace data be written to a trace table. PI trace entries contain information about program isolation ENQ/DEQ calls and DL/I calls.
- The PSB keyword requests a trace of all DL/I calls issued for a specified PSB.
- The TABLE keyword specifies that online tracing into the specified trace tables be started or stopped.

Use the CICS-supplied transaction CETR to trace DL/I activity. For DBCTL, CETR traces a DL/I request until it leaves DFHDBAT.

See "Trace entries produced by DBCTL" on page 132 for information on obtaining DBCTL trace entries. See *IMS Operator's Reference* for guidance on the syntax of /TRACE commands and keywords, and *IMS System Administration Guide* or *IMS Administration Guide: System* for guidance on the effects using /TRACE commands can have on your system.

Finding out current status of DBCTL activities

To find out the status of particular DBCTL activities, use the /DISPLAY command.

About this task

- The /DISPLAY command with the ACTIVE keyword gives you an overview of activity in the entire DBCTL subsystem including processing for BMPs and for threads processing scheduled CICS transactions. For each thread that is currently

active (has a PSB scheduled) from a CICS transaction, there is an entry "DBT" in the column headed "TYPE", as shown in the /DISPLAY command examples in *IMS Operator's Reference* . (The TYPE column shows the thread type and DBT stands for DBCTL thread.) The display may show fewer DBT threads than the number specified by MINTHRD in the DRA startup parameter table.

- The /DISPLAY command with the CCTL keyword displays all (or specified) CICS systems currently connected to DBCTL. To specify a CICS system, add a CCTLNAME, which is the APPLID of the connected CICS system. The /DISPLAY command with the CCTL keyword also displays the following items for all or specified CICS systems:
  - All in-doubts for a given CICS or for all CICS systems (when you enter /DISPLAY CCTL INDOUBT).
  - Pseudo recovery token (only when status is INDOUBT). See "Resolving indoubt CICS DBCTL units of work manually" on page 74 for information on using the pseudo recovery token in a /CHANGE command.
  - Recovery token.
  - Thread number (displayed as REGID) for all threads.
  - PSB name.
  - Status of thread(s).
  - All threads for a given CICS or all CICS systems.

  Note: The /DISPLAY command uses the CCTL ID (which, in the case of a CICS system, is the APPLID). However, many IMS messages use the jobname of the CICS system. Therefore, it is advisable to have a naming convention that enables operators to immediately identify a corresponding CICS APPLID and CICS JOBNAME. For example, if you use the APPLID DBDCICA, your job name could also contain the characters CICA.
- The /DISPLAY command with the OLDS keyword displays the system logging status. You can use it to determine how many OLDS data sets are available for use or require archiving.
- The /DISPLAY command with the POOL keyword displays main storage utilization statistics for IMS storage pools.
- The /DISPLAY command with the AREA keyword displays the status of DEDB data sets in an area.
- The /DISPLAY command with the DATABASE keyword displays the status (for example, NOTOPEN or STOPPED) of specified databases. If the database you specify is a DEDB, the associated DEDB areas are also displayed.
- The /DISPLAY command with the DBD keyword displays, for databases that are being accessed, their type, the PSBs accessing them, and the type of access. (You can use the DBD keyword only if you have DEDB support installed.)
- The /DISPLAY command with the MODIFY keyword displays the status of resources to be deleted or changed using the /MODIFY command. See "Changing DBCTL resources online" on page 57 for information on the /MODIFY command.
- The /DISPLAY command with the PSB keyword displays the status of PSBs, the databases being accessed, and the type of access. (You can use the PSB keyword only if you have DEDB support installed.)
- The /DISPLAY command with the PROGRAM keyword displays the status of PSBs; for example, NOTINIT or STOPPED.
- The /DISPLAY command with the SHUTDOWN STATUS keywords displays system activity during a shutdown type of checkpoint; for example, the number of regions still active.

- The /DISPLAY command with the STATUS keyword displays the status of DBCTL resources, such as databases and PSBs.
- The /DISPLAY command with the TRACE keyword displays status and options for IMS traces and the IMS monitor, and whether restart should occur without backout of BMP updates. (You can restart without using backout or recovery of databases: see the description of the COLDBASE keyword of the /ERESTART command in "Emergency restart" on page 67.)

Specifying messages to be logged on IMS log

Use the /LOG command to specify any alphanumeric character message to be logged on the IMS log.

Changing DBCTL resources online

The /MODIFY command is a part of the online change process used to control the modification of DBCTL resources online.

About this task

An online change for DBCTL is different from CICS resource definition online (RDO). You first use the offline process for doing a generation (whether it is an ACBGEN, or a partial MODBLKS generation for the DATABASE and APPLCTN macros). Guidance information about doing these generations is in *IMS System Definition Reference* or *IMS Installation Volume 2: System Definition and Tailoring* and *IMS Utilities Reference: Database*. To bring the new libraries online, use the /MODIFY command. First use the /MODIFY command with the PREPARE keyword to indicate the type of system definitions that must be replaced. Depending on the parameters entered, the system initiates quiescing of the appropriate resources. Then use the /MODIFY command with the COMMIT keyword to bring all newly defined resources online, update the changed resources, and invalidate the deleted resources. If the /MODIFY command deletes a database, the database is closed and made unavailable to programs. You cannot use the /MODIFY command on DEDBs.

If a failure occurs before a COMMIT completes, the changes defined by the /MODIFY command with the PREPARE keyword are not recovered across an emergency restart and you must reenter them. When a commit is successful, changes persist across all DBCTL restarts.

You can use the /MODIFY command with the ABORT keyword to reset the status that was set by the /MODIFY command with the PREPARE keyword. You can also use the /MODIFY command with the ABORT keyword if you have previously used the /MODIFY command with the COMMIT keyword, but it was not successful and you decide not to continue with the change. See also "Finding out current status of DBCTL activities" on page 55 for details of using the /DISPLAY command with the MODIFY keyword.

Preventing programs and transactions from updating DBCTL databases

You can use the /DBDUMP command with the DATABASE keyword to prevent programs from updating DL/I full function databases.

About this task

You can use the /DBRECOVERY command to prevent transactions or programs from accessing a database (with the DATABASE keyword) or a DEDB area (with the AREA keyword, which is valid with DEDBs only). The command closes and deallocates the database(s) or area(s), so that they are not authorized to DBRC.

If a specified database is being used when you enter either /DBDUMP or /DBRECOVERY, the thread currently using the database is allowed to complete, but no further PSB schedules are allowed.

If a database specified in either of these commands is being used by a BMP, an error message is issued, and the command is ignored for that database. You reenter the /DBDUMP or /DBRECOVERY command when the database is no longer being used by a BMP. If you need to recover the database immediately, use the /STOP command with the THREAD keyword (or its synonym, REGION) to terminate any BMPs using the database before you reenter the /DBDUMP or /DBRECOVERY command.

For a whole DEDB, the PSB is not scheduled. For a DEDB area, programs are not allowed access to data in that area. For a DL/I database, programs are not allowed access to the database.

Note: Issuing the /DBRECOVERY and /DBDUMP commands causes the OLDS to switch; an archive job may be generated to archive the previous OLDS. (This is controlled by the ARC=xx startup parameter.) Use the NOFEOV keyword to prevent the OLDS switching when you issue these commands.

The /START command reverses the effects of a /DBDUMP or /DBRECOVERY command. The /START command allocates the database or area. A database is authorized on the first schedule request it receives, and is opened at the first DL/I request. An area is authorized and opened on receipt of the first DL/I request.

Switching to a new OLDS

Specifying /SWITCH OLDS causes the IMS log to switch to the next OLDS. This switch to the next OLDS is marked as a recovery point for log archiving purposes. If you also specify the (optional) CHECKPOINT keyword, IMS issues a simple checkpoint after the active log data set has been switched to the next OLDS.

About this task

This switch capability is identical to that provided with the DBRECOVERY command, as described in “Preventing programs and transactions from updating DBCTL databases” on page 57 and “Log control with DBRC” on page 23.

Entering external subsystem commands from DBCTL

If you are using DBCTL to access DB2 databases via BMPs, you can use certain DBCTL operator commands to enter ***external subsystem*** commands (where DB2 is the external subsystem).

To display the status of all or specified external subsystems, use the /DISPLAY command with the SUBSYS keyword. (This is similar to using the /DISPLAY command with the CCTL keyword to display the status of CICS systems connected to DBCTL.)

To display the status of origin application schedule numbers (OASNs), which are IMS recovery elements in a DB2 subsystem, use the /DISPLAY command with the OASN and SUBSYS keywords. If you then need to purge any incomplete UOWs in the external subsystem, use the /CHANGE command with the SUBSYS, OASN, and RESET keywords.

To enter an external subsystem command from the DBCTL console or a program authorized do so, use the /SSR command. For example:

```
/SSR -DISPLAY THREAD
```

displays information about DB2 threads. The command is processed in DB2 and the response is sent back to the terminal from which you issued the /SSR command.

Making DBCTL resources available

To make DBCTL resources available to refer to and use, enter the /START command.

About this task

- Specify that the stopped status of particular DEDB areas be reset (AREA keyword).
- Change the automatic archiving option selected at system initialization or specified in a previous /STOP command (AUTOARCH keyword).
- Specify databases to be started so that they can be referenced by PSB schedule commands (DATABASE keyword).

  Add the NOBACKOUT keyword to the DATABASE keyword for databases that are not registered in DBRC and were backed out using standard batch backout. If your databases are registered with DBRC, the /START process inquires with DBRC whether backout needs to be done before starting a database.
- Specify that a previously stopped online log data set (OLDS) is to be started or that DBCTL is to add a new OLDS (OLDS keyword). (See "IMS online log data set (OLDS)" on page 22 for more information on this data set.)
- Specify a PSB to be started (PROGRAM keyword). DBCTL stops a PSB after most pseudo abend codes that can occur. If this happens, you must use a /START PROGRAM command before that PSB can be scheduled again.
- Start BMPs from a JCL partitioned data set (REGION keyword). Using /START REGION in this way enables you to keep all your BMP JCL in one place.
- Specify that a write-ahead data set (WADS) is to be added to the pool of WADS (WADS keyword).

Preventing scheduling of PSBs and use of DBCTL databases

You can use the /STOP command to stop the scheduling of specific PSBs and the use of a given database.

About this task

The /STOP command works as follows:

- The ADS keyword specifies that a DEDB area data set (ADS) is to be stopped and deallocated. Note that this command stops only the ADS, not the entire area. The area is stopped only if there is no ADS allocated. This command is

rejected if the ADS you specified is the last data set available in the area because ADSs are invalidated when they are stopped. ADSs are reestablished by running the DEDB area data set create utility.

- The AREA keyword specifies that all the data sets associated with an area are to be stopped and deallocated. The status of this area is set to STOP, as displayed with a /DISPLAY DATABASE command. (See "Finding out current status of DBCTL activities" on page 55.) If the area is already stopped, the /STOP command just deallocates the data sets.
- The AUTOARCH keyword specifies that automatic archiving is to be stopped.
- The DATABASE keyword stops the use of the specified database.
- The OLDS keyword specifies that DBCTL is to stop using an OLDS.
- The PROGRAM keyword specifies that a PSB is to be stopped.
- The REGION or THREAD keywords specify a region or thread that is to be stopped. This can be a region or thread shown by the /DISPLAY CCTL command. (See "Finding out current status of DBCTL activities" on page 55.)
- The WADS keyword indicates that a WADS is to be removed from the pool of WADS.

Purging a transaction that is using DBCTL

You can query and purge tasks that use DBCTL using the CICS CEMT transaction as for any CICS task.

About this task

However, if a transaction has "hung" in DBCTL, and you need to purge it, you must use the DBCTL command /STOP THREAD.

To find out what is happening to a task:

1. Issue CEMT INQ TASK to find out what tasks are active.
2. Expand the information on individual tasks by typing a ? to the left of the task you want to see. You will get a display like the one in Figure 23 on page 61.

```
I TA
SYNTAX OF SET COMMAND
 Tas(0000110) Tra(DLID) Fac(D2D3) Sus Ter Iso Pri( 001 )
    Hty(DBCTL   ) Hva(DLSUSPND) Hti(000007) Sta(TO)
    Use(CICSUSER) Rec(X'9EDA1F61E11CFA02')
 CEMT  Set TAsk() | < All >
  < PRiority() >
  < PUrge | FOrcepurge >









                                                SYSID=CIC1 APPLID=DBDCCICS

PF 1 HELP       3 END                   7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

Figure 23. CEMT INQ TASK (expanded)

Figure 23 includes the following useful information:

- `Tas(0000110)`: task identifier
- `Tra(DLID)`: transaction name of the task
- `Fac(D2D3)`: identifier of the terminal or queue that initiated the task
- `Sus`: the task is suspended
- `Ter`: the task was initiated from a terminal
- `Pri(001)`: the task is running with a priority of 1
- `Hty(DBCTL)`: the task is currently issuing a DL/I request to DBCTL
- `Hva(DLSUSPND)`: the task is suspended in DBCTL
- `Hti(000007)`: how long, in seconds, the task has been sus pended
- `Sta(TO)`: how the task was started; TO means from a terminal by an operator entering a transaction
- `Use(CICSUSER)`: is the userid of the user who initiated the task
- `Rec(X'9EDA1F61E11CFA02')` shows the recovery token associated with the task
- The screen also contains a reminder of the syntax of the CEMT SET TASK command, which you may need to use; for example, if you want to purge the suspended task.
- `SYSID=CIC1`: CICS system identifier, as specified in the system initialization parameter SYSIDNT.
- `APPLID=DBDCCICS`: APPLID for the CICS system.

3. Issue CEMT INQ TASK again.
   - If the response indicates that the task is no longer suspended in DBCTL, you can purge it using CEMT SET TASK(n) PURGE as for any CICS task. The purge takes place after the DL/I request to DBCTL has completed.
   - If the response indicates that the task is still suspended in DBCTL, the task has “hung” in DBCTL, and you must use DBCTL operator commands to purge it.

To do this:

1. From the CEMT INQ TASK display, make a note of the CICS APPLID and the 16-digit recovery token. (You can use a recovery token to find the thread number of a CICS task in DBCTL. For a fuller definition, see "CICS DBCTL recovery tokens" on page 73.)
2. At the DBCTL console, enter /DISPLAY CCTL cctlname, where cctlname is the CICS APPLID (in this example, it is DBDCCICS). This causes the current status of DL/I activity to be displayed, as shown in Figure 24.

```
0080  /DIS CCTL DBDCCICS
0080  DFS000I MESSAGE(S) FROM ID=SYS1 047
0080      CCTL     PSEUDO-RTKN  RECOVERY-TOKEN   REGID  PSBNAME   STATUS
0080      DBDCCICS                                          ATTACHED
0080                           9EDA1F61E11CFA02     6  PC3COCHD  ACTIVE
0080                           9EDA1F4E9B571B02     5  PC3COCHD  ACTIVE
0080      *88204/101241*
```

Figure 24. Output from /DISPLAY CCTL cctlname

3. Find the recovery token (9EDA1F61E11CFA02 in this example) that matches the one you noted from the CEMT INQ TASK display, and then note the thread number that is next to it in the REGID column (6 in this example).
4. Issue the command:

   ```
   /STOP THREAD n ABDUMP
   ```

   where n is the thread number.

   This causes the thread and transaction to terminate when it has finished processing the current request, and causes a dump to be taken.

   If the thread does not stop, use:

   ```
   /STOP THREAD n CANCEL
   ```

   Do not use /STOP THREAD CANCEL if you do not need to, because it may cause DBCTL to terminate with a U113 abend.

Stopping DBCTL normally

To stop DBCTL normally and disconnect it from CICS, use the /CHECKPOINT command with the FREEZE or PURGE keywords.

About this task

Active threads are terminated, CICS threads are terminated when they reach a sync point, and BMPs are processed until they reach a checkpoint, a SYNC call, or the end of a program. Shutdown then completes and the system status is saved in a system checkpoint on the log, and in the checkpoint ID table on the restart data set.

The difference between the FREEZE and PURGE keywords applies to BMPs. FREEZE stops them after the next checkpoint, or at program completion, whichever is the sooner, and PURGE allows them to complete.

When you have stopped DBCTL using /CHECKPOINT FREEZE or /CHECKPOINT PURGE, you can warm start it using /NRESTART, as described in "Warm start" on page 67.

Stopping DBCTL abnormally

There is no equivalent of a CICS immediate shutdown in DBCTL. If you need to force termination of DBCTL, the MVS console operator has to issue an MVS MODIFY jobname STOP command.

About this task

This causes an abnormal termination without a dump. If you want a dump to be taken, use an MVS MODIFY jobname DUMP command. For guidance on using MVS commands with IMS, see *IMS Operator's Reference* .

Dealing with messages from DBCTL and CICS

Messages from DBCTL (in the form DFSnnnn) are sent to console(s) as specified in the MCS= parameter of the IMSCTRL macro in the IMS generation. These messages include notification of change in status and of abnormal events.

About this task

There are many additional messages in the DBCTL environment. You can direct them to the console from which DBCTL commands will be entered. However, if you find that the volume of messages means it is impractical to view them "live" at the console, you may prefer to direct them to the console log and process them with whatever tool your installation uses to review console output.

The DFS554 message is a notification of the abnormal termination of a BMP region or a thread from a CICS transaction. If it has been caused by an abnormal termination of a thread that originated from CICS, the message text contains the CICS job name or CICS started procedure name. It also contains the abend code in the form SSS, UUU where SSS is a system abend code and UUU is an IMS user abend code. (See "Return codes in DBCTL" on page 137 for more information on these codes.) The message may contain the characters "PSB." If it does, the PSB contained in the message has been stopped. All attempts to schedule that PSB will fail until a /START PROGRAM command is issued for that PSB. See *IMS Messages and Codes* for guidance on interpreting DFSnnnn messages.

Messages from CICS that relate to DBCTL (for example, those relating to the CDBC transaction) are sent to the transient data destination CDBC so that they are located in one place. You can reroute these messages from CDBC, as you can with CSMT.

You can ***suppress or reroute*** messages sent to transient data queues such as CDBC. You can reroute from CDBC to a list of consoles, from CDBC to a different transient data queue, or reroute console messages from their transient data queues to CDBC. For programming information about coding the CICS-supplied user exit used to re-route messages and on the example user exit provided to help you do so, see *CICS Customization Guide*.

Messages DFHDB8103 and DFHDB8104 are issued if there is a failure to connect to DBCTL. They contain the DBCTL reason codes for the connection failure.

Message DFHDB8109 is issued when:

- A schedule request has failed.
- DBCTL has abnormally terminated a thread and, as a result, CICS abnormally terminates the transaction.

Message DFHDB8109 is ***not*** issued when an error type status code is returned to the application program.

DFHDB8109 enables you to identify the IMS reason for which this CICS transaction has failed. The *IMS Messages and Codes* contains guidance on interpreting the IMS abend and reason codes referred to above.

Chapter 4. Recovery and restart operations for DBCTL

Covers recovery and restart, commit protocols and resolving indoubt units of work.

Overview of CICS and IMS recovery and restart

CICS and IMS perform similar recovery functions, but there are differences in terminology and in implementation.

See *IMS Operations Guide* for background information on recovery in IMS. If you are familiar with CICS or IMS, but not both, read this overview and then read the manual for the product that you are not familiar with.

CICS startup and shutdown

CICS has different types of startup and shutdown and these affect the DBCTL connection.

CICS has the following types of initialization or restart depending on the START system initialization parameter and on how CICS was last terminated:

- Initial start
- Cold start
- Warm start
- Emergency restart.

You cannot specify warm start or emergency restart explicitly. Instead, you specify the START=AUTO system initialization parameter, and CICS determines which of these two kinds of start to use.

If CICS performs a warm start or an emergency restart on a system to which DBCTL was connected and DBCTLCON=YES is specified as a system initialization parameter, the same DRA startup table suffix is automatically used when DBCTL is reconnected. The suffix might change if you have used the **INITPARM** system initialization parameter, as described in "Reviewing CICS system initialization parameters" on page 12, to override the suffix previously used. For information on methods of connecting to the same, or a different, DBCTL see "Connecting DBCTL to CICS automatically" on page 36.

CICS initialization begins when the job is submitted and, in almost all cases, continues until completion of the specified type of restart. Error conditions might require operator replies or might cause abnormal termination.

CICS has three types of termination:

- Normal
- Immediate
- Abnormal, due to abend or an MVS CANCEL

The CICS master terminal command to shut down CICS has two options: normal and immediate. A normal shutdown allows transactions to complete before shutting down and saves the system status in the CICS catalog. You can do a

warm start after a normal shutdown. An immediate shutdown does not allow transactions to complete. It is equivalent to an abnormal termination, and you must restart CICS using emergency restart.

There are special considerations for canceling CICS when it is connected to DBCTL. See the information on causing an abnormal termination of CICS, in "CICS failure" on page 78.

Restarting DBCTL

DBCTL has three types of (re)start:

- Cold (/NRESTART CHECKPOINT 0)
- Warm (/NRESTART)
- Emergency (/ERESTART)

The startup process has two distinct phases: initialization and restart. You can use AUTO restart to do either a warm start or an emergency restart.

With an AUTO restart, (DBCTL startup parameter AUTO=Y), DBCTL decides whether warm start or emergency restart is required, based on the contents of the IMS restart data set (RDS), and proceeds with the restart without your needing to enter any further restart command.

If you need to enter your own restart command (for example, to perform a cold start), use a non-AUTO restart (DBCTL startup parameter AUTO=N). Non-AUTO restart stops after initialization, at which point you must manually enter a restart command.

AUTO=N will have been specified, or defaulted to, for the first startup of DBCTL. For subsequent restarts, use warm start or emergency restart, which means that you will need to change the parameter to AUTO=Y. For guidance on specifying AUTO=Y and AUTO=N, see *IMS System Definition Reference* or *IMS Installation Volume 2: System Definition and Tailoring*.

During restart processing, the log and RECON are opened.

The sections that follow state how you use these types of (re)start with DBCTL.

Cold start

With this type of start, DBCTL is brought up in the state it was in at system generation.

Do not use cold start after a DBCTL failure. Instead, use an emergency restart. See "Emergency restart" on page 67 for more information.

To request a cold start of DBCTL, use the /NRESTART command with the CHECKPOINT 0 keyword. Additional keywords with /NRESTART CHECKPOINT 0 enable you to:

- Specify whether you want the RDS, or the WADS (or both) formatted as part of restart process (the RDS, WADS, or ALL keywords). Format the RDS and the WADS if there has been a data set I/O error, if you need to reallocate a data set or change its size, or if you are starting DBCTL for the first time.
- Specify whether the IMS system definition password security option is to be in effect: provided your system definition enables operators to change password security (the PASSWORD keyword).

Before you do a cold start, you must ensure that the IMS you intend to start does not have a subsystem record in the RECON. This will be the case if it is a new subsystem, if it was shut down normally the last time it was used, or if it was not shut down normally but the appropriate DBRC commands (including DELETE.SUBSYS) and other actions needed to ensure database integrity were performed.

Warm start

With this type of start, DBCTL is brought up in the environment it was in when it terminated normally using a /CHECKPOINT FREEZE or /CHECKPOINT PURGE command.

This is described in "Stopping DBCTL normally" on page 62. After a warm start, resources are in the same state they were in at the time the system was shut down.

The difference between the FREEZE and PURGE keywords applies to BMPs. FREEZE stops them after the next checkpoint, or at program completion, whichever is the sooner, and PURGE allows them to complete. See *IMS Operator's Reference* for a list giving guidance on the differences between these options.

To request a warm start of DBCTL, use the /NRESTART command without CHECKPOINT 0.

Any indoubt UOWs are re-created for this type of start. (An ***indoubt*** UOW is a piece of work that is pending during commit processing. If commit processing fails between DBCTL's response to CICS's request to prepare for commit and CICS's decision to execute the commit, recovery processing must resolve the status of any work that is indoubt.) See "Resolving indoubt CICS DBCTL units of work manually" on page 74 for information on using operator commands to resolve indoubt UOWs.

You can use the following optional keywords on /NRESTART:

- If the WADSs have been reallocated, specify whether you want them to be formatted as part of the restart process. Format the RDS and the WADS if there has been a data set I/O error or if you need to reallocate a data set or change its size.
- Specify whether the IMS system definition password security option is to be in effect: provided your system definition enables operators to override password security.

Emergency restart

With this type of start, DBCTL is restarted in the environment it was in ***before*** a DBCTL failure.

To perform an emergency restart of DBCTL, use the /ERESTART command. DL/I in-flight UOWs (that is, those that were still being processed when the failure occurred) are backed out. Committed but unwritten DEDB changes are applied to the database. Units of work that were indoubt are retained and are resolved automatically when CICS and DBCTL are reconnected. For further guidance on how this is done, see *IMS Operations Guide*. If the UOWs fail to be resolved automatically, you can use DBCTL operator commands to do so, as described in "Resolving indoubt CICS DBCTL units of work manually" on page 74.

If a failure in emergency restart prevents backout being completed, instead of using a COLD start, you can reattempt the emergency restart using the COLDBASE keyword on the emergency restart command. Full function DL/I

databases and DEDB areas that have indoubt data or that need backout or recovery are identified and stopped. Database backout and committed DEDB updates are not done. You must then use the appropriate IMS utilities to backout or forward recover these databases. (See *IMS Utilities Reference: Database* for guidance on using the utilities.)

You can also specify whether the restart or write ahead data sets should be formatted as part of the restart process. Format the RDS and the WADS if there has been a data set I/O error or if you need to reallocate a data set or change its size.

CICS keypoints and IMS checkpoints

This section discusses system-level keypoint and checkpoint information. Both CICS and IMS also have task or program (thread) level synchronization information.

CICS keypoints and IMS checkpoints both contain system status information that is modified during online operation. The concepts are basically the same, but they are implemented differently.

A CICS warm start uses a warm keypoint that was written to the CICS catalog by the previous normal CICS shutdown.

A CICS emergency restart reads the CICS system log backwards until it has located an activity keypoint. The keypoint contains a record of incomplete UOW chains which CICS reads directly. These chains can reside on the primary and secondary system logs.

An IMS warm start reads the checkpoint ID table on the RDS to find the shutdown checkpoint on the log. The RDS is a data set that IMS uses to record system checkpoint ID information during the logging process. IMS finds the information it needs and uses it automatically. If the RDS is not available at restart, you can obtain the checkpoint information needed from the log, but this may lengthen the restart process. Generally, you do not need to know the content of the RDS. However, if you are faced with a particularly complex recovery problem, you may need to examine the RDS. You can find guidance on its contents in *IMS Operations Guide*.

An IMS emergency restart reads the checkpoint ID table on the RDS and selects the checkpoint that precedes the last synchronization point of each program that was active at the time of the failure. It then reads the IMS log forward from the selected checkpoint.

To take a simple checkpoint of DBCTL, use the /CHECKPOINT command.

Backing out uncommitted updates after a failure

The meaning of the term ***dynamic backout*** differs slightly between CICS and IMS.

In CICS, dynamic backout means backout as a result of a transaction (or application program) failure. The term **transaction backout** is used for backout done during CICS emergency restart.

In IMS, dynamic backout means backout as a result of a program failure. In a DBCTL environment, program failures include CICS transaction abends and BMP failures. The IMS /ERESTART command also performs emergency restart backout. IMS provides a batch backout utility, DFSBBO00, which you can use if dynamic

backout or emergency restart fails. See *IMS Operations Guide* for guidance on ***when*** to run this utility, and *IMS Utilities Reference: Database* for guidance on ***how*** to run it.

Because IMS does the backing out of database updates in a DBCTL environment, we concentrate on IMS backout in this section.

For IMS full function databases, database changes are placed in the log buffers and the database buffers as they are made. Depending on system activity, they may be written before they are committed and so, after a program failure or an IMS system failure, databases may require backout. The IMS log data sets (OLDS) are used for dynamic backout. (See "IMS online log data set (OLDS)" on page 22 for more information.) Additionally, if dynamic backout or /ERESTART backout fails, for a database, that database is stopped. The backout is automatically reattempted when the database is restarted.

For DEDBs, no changes are placed in the log buffers until syncpoint processing begins, and no changes are written to the database until a commit has been received. This means that they do not need backout if there is a failure during phase 1 of the syncpoint process. The system can undo the changes by releasing the database buffers that have been modified but not yet written.

Log records

The IMS log is a record of activities and database changes. Among the log records written to the IMS log are those that record both phases of the commit for each unit of work.

These log records contain the information necessary for database recovery and system restart. The *IMS Diagnosis Guide and Reference* contains, for guidance, a list of the types of log records and tells you how to obtain a listing of these DSECTs. The *IMS Utilities Reference: Database* gives guidance on using the file select and formatting print utility, DFSERA10, to print the IMS log records.

Database recovery control (DBRC)

Database recovery control (DBRC) assists you in controlling DBCTL logs, and in managing recovery of databases.

With DBCTL, you ***must*** use DBRC to control your logs, and you may optionally use it to control batch logs and database recovery. DBCTL requires DBRC to be at SHARECTL level; if it is not, DBCTL will not start.

You may optionally use DBRC to control the data sharing environment by allowing (or preventing) access to databases by various subsystems sharing those databases.

If you use DBRC to control database recovery, you must register your databases with DBRC, so that it can record the relevant information in the RECON, and then use that information to control the recovery of your databases. See *IMS Operations Guide* for general guidance on registering databases. You can register your databases using either of the following:

- The recovery control utility, DSPURX00. See *IMS Utilities Reference: Database* for guidance on using DSPURX00.
- The /RMINIT.db and /RMINIT.dbds commands. See *IMS Operator's Reference* for guidance on the syntax of these commands.

To recover a database that is registered with DBRC, use the /RMGENJCL.RECOV command. DBRC recovers the database using a combination of available input; for example, image copy data set, change accumulation data sets, log data sets, and archived log data sets.

Recovery control (RECON) data sets

DBRC automatically records information in dual recovery control (RECON) data sets. Both data sets contain identical information, and so are usually referred to as one: the RECON. The information from the RECON is needed during warm and emergency restarts. DBRC selects the correct data sets to be used by a recovery utility when you enter a GENJCL command. For a restart, the RECON shows which data set, the OLDS or the SLDS, contains the most recent log data for each database data set (DBDS) you have registered with DBRC. For the OLDS, the RECON shows whether the OLDS has been closed and whether it has been archived. The RECON contains timestamp information for each log data set and volume. IMS uses this information to determine which data set and volume contain the checkpoint information needed to restart DBCTL.

Commit protocols and units of recovery for DBCTL

This section describes what happens when a transaction has updated DBCTL databases, and is issuing a syncpoint, or a TERM request, or is terminating. If a failure occurs at any of these stages, DBCTL might not be able to determine whether CICS intended these updates to be backed out or committed and must request this information from CICS when it has been reconnected.

Two-phase commit for DBCTL

DBCTL uses a ***two-phase commit*** to record a syncpoint. At the completion of a two-phase commit, the requested processing is committed and if a failure occurs, DBCTL does not ABORT committed changes.

Two-phase commit consists of the PREPARE and COMMIT phases. Within the PREPARE phase, CICS issues a PREPARE request to DBCTL. DBCTL writes to the log and issues its response to the PREPARE request to CICS. Within the COMMIT phase, there are two possible actions: COMMIT and ABORT. The ABORT action for data belonging to full function DL/I databases is ***backout***. There is no backout for data belonging to DEDBs because, as explained below, it is not written to the database before the COMMIT phase. The effect of an ABORT for DEDBs is also referred to as ***undo***. Because a CICS thread may be accessing data belonging to both full function DL/I databases and DEDBs, we use the term ABORT to refer to both backout and undo.

When updates are written to databases

The DEDB terms UNDO and REDO are analogous to the DL/I full function terms BACKOUT and COMMIT respectively. However, although the processes that these terms refer to have the similar end results, the processes themselves differ.

The difference is in the stage at which updates are written to the database. This is shown in Figure 25 on page 71.

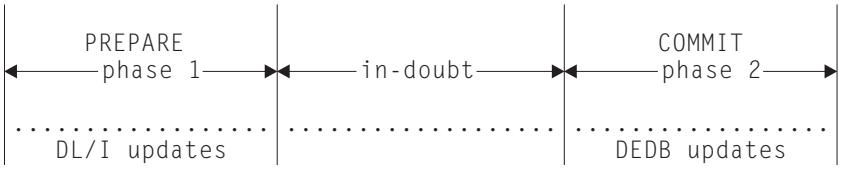


Figure 25. When updates are written to databases

This difference in timing of writing updates dictates the action taken during the second phase of two-phase commit.

For full function DL/I databases:

- If the phase 2 action is COMMIT, no action is needed to commit updates because DL/I wrote them to the database during phase 1.
- If phase 2 action is ABORT, a BACKOUT of the updates is required because DL/I wrote them to the database during phase 1.

For DEDBs:

- If phase 2 action is COMMIT, the changes must be REDOne to the database because they have only been made in main storage. (They are written (committed) to the database on DASD by the output threads, which are generated by the IMS system generation parameter OTHREADS. See *IMS System Definition Reference* or *IMS Installation Volume 2: System Definition and Tailoring* for guidance on this parameter.)
- If phase 2 action is ABORT, no changes have to be made to the database, because the changes are still in main storage, and can be UNDOne from there.

REDO is also used to refer to the action required for committed DEDBs during emergency restart of IMS. IMS can determine from the log that a COMMIT was initiated, but that phase 2 is not indicated as complete. In this case, DEDB updates must be REDOne. The two phases are:

1. Phase 1, in which CICS directs syncpoint preparation and asks whether or not the updates to DBCTL databases can be committed.
2. Phase 2, in which CICS tells DBCTL that it must either COMMIT or ABORT the resources. (CICS can request an ABORT without first issuing a PREPARE request. That is, CICS can bypass the first phase of two-phase commit when an update is being backed out.)

Figure 26 on page 72 shows two-phase commit and describes the activities taking place.

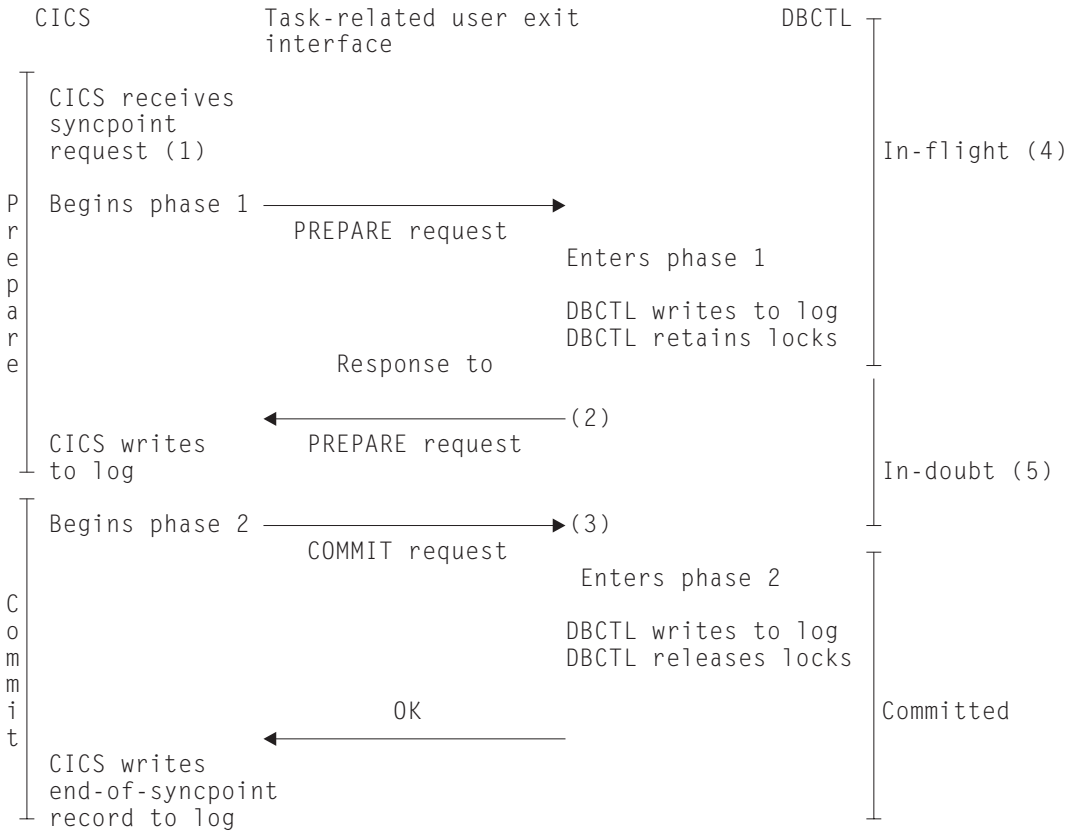


Figure 26. Two-phase commit

Note:

1. The syncpoint request can be EXEC CICS SYNCPOINT, a DL/I TERM call, or a CICS task termination.
2. If DBCTL indicates that it cannot commit the updates, CICS aborts the unit of recovery and the rest of the Figure 26 does not apply.
3. If CICS tells DBCTL to commit the updates, DBCTL ***must*** commit.
4. At this stage, units of recovery are ***in-flight*** and, if DBCTL fails, all database updates are aborted.
5. At this stage, from the time that DBCTL issues its response to the PREPARE request to the time it receives a COMMIT request from CICS, units of recovery are ***indoubt***. DBCTL retains the indoubt information. When DBCTL is restarted after a failure, it inquires with CICS about the status of the in-doubts. This is part of ***resynchronization***.

UOWs and resources belonging to multiple resource managers

The two-phase commit process also applies if a UOW is updating resources that belong to more than one resource manager; for example, any of the following: DBCTL databases (DL/I full function or DEDBs, or both), local VSAM files, and DB2 databases. As explained above, CICS is the coordinator of the two-phase commit process; DBCTL is a participant. CICS must ensure that all the resource managers, including DBCTL, are in synchronization. To do this, at phase 1 of two-phase commit, it issues a PREPARE request to all the resource managers involved to find out if a COMMIT can be done. This is as shown in Figure 26, in which CICS is communicating with DBCTL only. If ***all*** the other resource managers indicate that a COMMIT is possible, CICS tells them all to COMMIT. If not, CICS tells them ***all*** to ABORT. The COMMIT or ABORT must now be carried out in all

the resource managers. For this reason, CICS considers the COMMIT or ABORT to be completed at this stage, even if it is slightly delayed.

DBCTL unit of recovery

A DBCTL **unit of recovery** is created for each processing request when the first schedule request is made by the transaction, and is kept until the two-phase commit is complete. As described in "Resolving indoubt CICS DBCTL units of work manually" on page 74, commands are available to display the units of recovery and take appropriate actions for committing or ending them.

In-flight unit of recovery

If DBCTL fails and is subsequently restarted, all in-flight units of recovery are backed out.

Indoubt unit of recovery

When a failure occurs, a recoverable indoubt structure (RIS) is constructed for each indoubt unit of recovery and is also written to the IMS log. The RIS contains:

- Residual recovery element (RRE), which contains the recovery token.
- Indoubt extended error queue element (IEEQE), which contains the changed data records.
- Buffer extended error queue element (BEEQE), which indicates a data block that cannot be accessed because of unresolved in-doubts.
- Extended error queue element link (EEQEL), which links the basic portion of the RIS (the RRE) with the IEEQE and the BEEQE, which are used to protect indoubt data.

The IMS batch backout utility, DFSBBO00, and the IMS database recovery utility, DFSURDB0, process the indoubt units of recovery.

CICS units of work (UOWs)

CICS UOWs and DBCTL units of recovery are more or less synonymous, except that from CICS's point of view, a UOW begins at the beginning of a task, and a unit of recovery begins when that task issues its first DL/I request. For simplicity, in the rest of this book, we use the CICS term UOW to refer to both. The IMS publications use the term "unit of recovery".

CICS DBCTL recovery tokens

Recovery tokens are created by CICS and passed to DBCTL. They are unique identifiers for each UOW. The lifetime of a recovery token is the same as for a UOW.

You can use them to correlate work done between CICS and DBCTL in the same UOW. Each recovery token is 16 bytes long; the first 8 bytes are the CICS APPLID (passed to DBCTL when CICS is first connected) and the second 8 bytes are a UOW identifier. CICS creates an identifier like this for every UOW. DBCTL validates the recovery token to protect against duplication of UOWs. You can use the recovery token in certain operator commands. For example, you can display it as part of the output of the /DISP CCTL and CEMT INQ TASK commands, and you can enter it in /CHANGE commands, in the form of a pseudo recovery token, as explained below. The recovery token is included in certain messages (for example, the CICS message DFHDB8109, which is issued when a DL/I request has failed). Recovery tokens can be useful in problem determination, because they are

displayed in dumps produced by CICS and DBCTL and in trace entries produced by CICS. See Chapter 7, “Problem determination for DBCTL,” on page 119 for more information.

The pseudo recovery token is an 8-character decimal token, which can be used in place of the 8-byte hexadecimal recovery token and is displayed when the status of a thread is indoubt. It is made shorter than the recovery token so that it is easier to make note of (for example, from /DISPLAY commands) and enter (for example, in /CHANGE commands).

Figure 27 shows a pseudo recovery token (00010040 in the column headed PSEUDO-RTKN) and a recovery token (F0F58879641002C2) for thread number 4 (in the column headed REGID) for PSBNAME PC3COCHD, whose STATUS is INDOUBT.

```
0080  /DIS CCTL DBDCCICS
0080  DFS000I MESSAGE(S) FROM ID=SYS1 047
0080      CCTL     PSEUDO-RTKN  RECOVERY-TOKEN   REGID  PSBNAME   STATUS
0080      DBDCCICS                                        ATTACHED
0080                           9EDA1F61E11CFA02     6  PC3COCHD  ACTIVE
0080                           9EDA1F4E9B571B02     5  PC3COCHD  ACTIVE
0080               00010040    F0F58879641002C2     4  PC3COCHD  INDOUBT
```

Figure 27. /DISPLAY CCTL cctlname command showing pseudo recovery token

Resolving indoubt CICS DBCTL units of work manually

Normally, an emergency restart of DBCTL followed by reconnection of CICS and DBCTL after a failure should resolve in-doubts automatically.

About this task

However, you may sometimes need to do this yourself. For example, if a CICS system using DBCTL disconnects abnormally from DBCTL (for instance, if CICS or DBCTL abends, or CDBC DISCONNECT IMMEDIATE is issued), there may be some incomplete updates about which DBCTL is in doubt. Even if CICS then needs to be cold started for some reason, it normally recovers enough information to resolve indoubts automatically. However, if CICS is started with the START=INITIAL system initialization parameter, it loses its record of the indoubt updates and they must be resolved manually. You are strongly advised not to start CICS with START=INITIAL specified when there are indoubt units of work outstanding.

The DFS2283I message, issued during the resynchronization process, indicates that there are UOWs that have not received a COMMIT or ABORT request, and are therefore indoubt.

In this situation you must use DBCTL operator commands (described in “Using DBCTL operator commands to resolve in-doubts”) to resolve the in-doubts.

Using DBCTL operator commands to resolve in-doubts

Use the following DBCTL operator commands to commit or backout a unit of work.

1. Use /DISPLAY CCTL cctlname INDOUBT, as shown in Figure 28 on page 75 to obtain the pseudo recovery token that identifies the indoubt work. (Pseudo

recovery tokens are defined in “CICS DBCTL recovery tokens” on page 73.)

```
0080  /DIS CCTL DBDCCICS INDOUBT
0080  DFS000I MESSAGE(S) FROM ID=SYS1 047
0080      CCTL     PSEUDO-RTKN  RECOVERY-TOKEN   REGID  PSBNAME   STATUS
0080      DBDCCICS                                                 ATTACHED
0080               00010040     F0F58879641002C2     4  PC3COCHD  INDOUBT
```

Figure 28. /DISPLAY CCTL cctlname command showing indoubt

2. Use /CHANGE CCTL cctlname PRTKN token command to abort or commit the indoubt. The cctlname is the APPLID of the CICS system. The PRTKN keyword specifies the pseudo recovery token of the element to be processed. The command is either:
   - ABORT to backout changes for a unit of recovery, or COMMIT to commit changes for recovery. For example:

     ```
     /CHANGE CCTL DBDCCICS PRTKN 00010040 COMMIT
     ```

     would commit the indoubt shown in Figure 28.

When the action you specified has been completed, the recoverable indoubt structure (RIS) for the indoubt UOW is removed.

IMS database utilities

DBCTL enables you to use utilities that IMS provides to help with the backup and recovery of your databases.

These utilities are listed below.

Note: Because database change records are written to the IMS log, you do not need to retain the CICS system log for use by IMS database recovery utilities in a DBCTL-exclusive environment.

- Database image copy utility, DFSUDMP0

  The database image copy utility, DFSUDMP0 is a batch utility that creates a copy of data sets within a database. For DEDBs, you can copy an area concurrently with DBCTL activity. You can also use concurrent image copy for full function DL/I databases.

  If the databases are updated while the utility is running, all logs including the one that was being used when DFSUDMP0 was started, are needed for use with DFSURDB0. You need both the log and the image copy to give a complete “picture” of the database for recovery purposes.

  If you have not created an image copy, the data set to be recovered is used as input to DFSURDB0.
- Online database image copy utility, DFSUICP0

  The online database image copy utility, DFSUICP0, is a BMP that creates an output copy of a data set within a full function DL/I database while the database is allocated and being used by DBCTL.

  If the databases are updated while the utility is running, all logs including the one that was being used when DFSUICP0 was started, are needed for use with DFSURDB0. You need both the log and the image copy to give a complete “picture” of the database for recovery purposes.

  If you have not created an image copy, the data set to be recovered is used as input to DFSURDB0.

- Database change accumulation utility, DFSUCUM0

  If system availability is a major concern for your installation, you will probably want to use this utility. It collects the changes from the other log data sets onto a single log, thus helping to speed recovery. Balance the benefits of using it against the overhead it incurs, and the fact that you may not need to use its output.
- Database recovery utility, DFSURDB0

  The database recovery utility uses a backup copy of your database together with either (or both) the change accumulation utility or the logs, and reapplies changes made since the backup copy to create a new, reconstructed, database.

  The database recovery utility performs recovery at the data set level, or at the track level. Often, only a single data set of the database requires recovery. However, if more than one data set has been lost or damaged, you need to recover each one separately. If an I/O error caused the problem, you might need to recover only a single track instead of reconstructing the entire data set.

You can use these utilities together to perform recovery by updating a copy of the database with the changes logged since the copy was made, as shown in Figure 29 on page 77. See *IMS Utilities Reference: Database* and *IMS Operations Guide* for further guidance on using the utilities, including any restrictions that may apply.

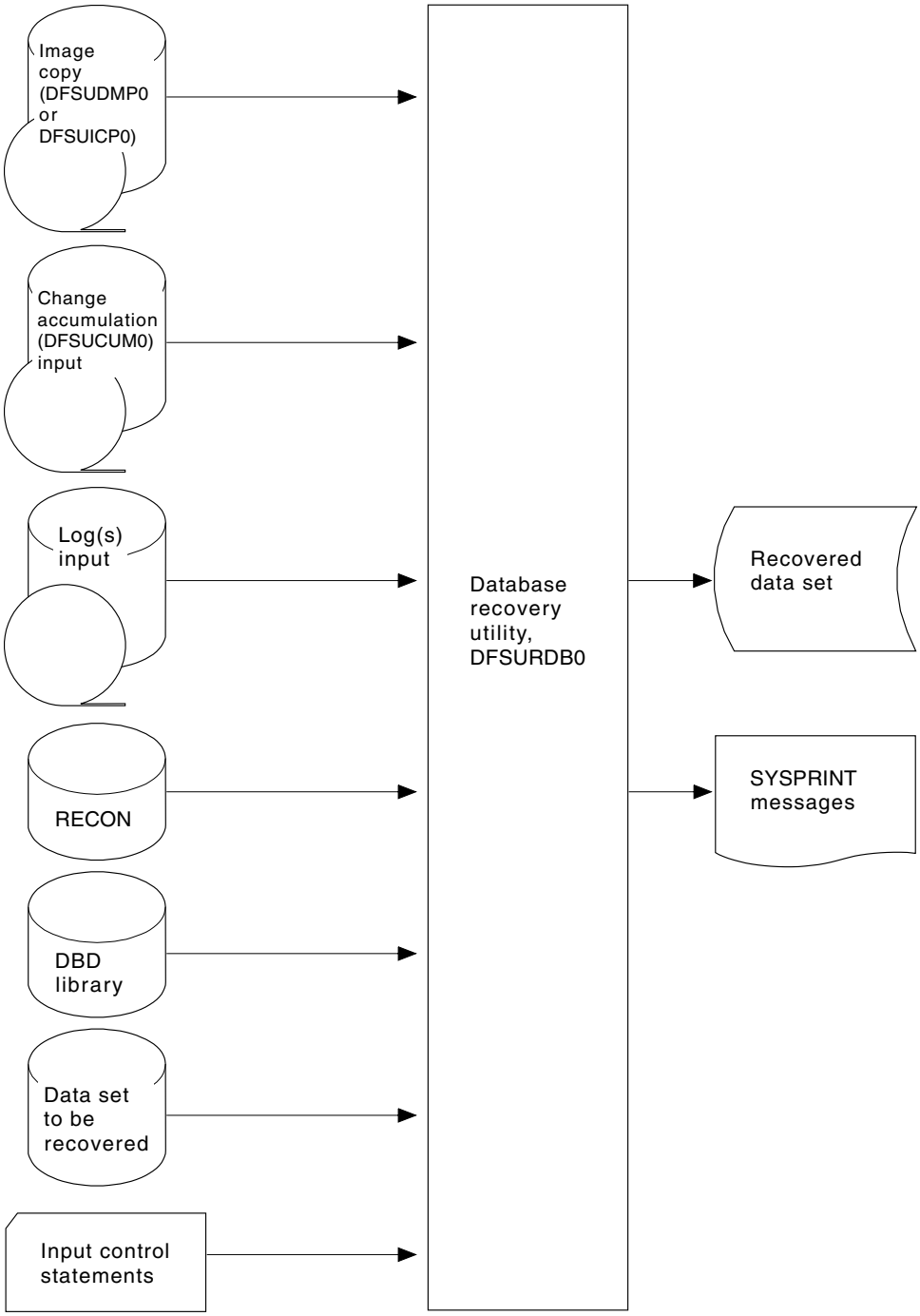


Figure 29. Database recovery utility, DFSURDB0, showing inputs and outputs

Note: Input from the image copy and change accumulation utilities is optional.

IMS log utilities

DBCTL enables you to use the following IMS log utilities: the log archive utility, the log recovery utility, and the file select and print formatting utility.

The IMS log utilities are as follows:

- The log archive utility, DFSUARC0 produces a system log data set (SLDS) from a filled OLDS. DBCTL can automatically invoke DFSUARC0 to archive the OLDS when an OLDS switch occurs. You use the ARC= parameter in the DBC procedure to control automatic archiving. See *IMS System Definition Reference* or *IMS Installation Volume 2: System Definition and Tailoring* for further guidance on specifying ARC, and *IMS Utilities Reference: Database* for guidance on setting up the skeleton JCL needed. Alternatively, you can use the DBRC command GENJCL.ARCHIVE to initiate manually an archive if you did not specify the automatic archive option, or if an automatic archive fails. See *IMS Operations Guide* for further guidance about automatic archiving. The log archive utility runs as a batch job, and you can run multiple log archive jobs concurrently. The SLDS it creates can be on DASD, MSS, or tape. DFSUARC0 is the preferred utility for archiving logs in a CICS-IMS environment.
- The log recovery utility, DFSULTR0 produces a usable log data set from one that contains read errors or could not be closed properly. You can recover both system log data sets (SLDSs) and online log data sets (OLDSs) with this utility.
- The file select and print formatting utility, DFSERA10 enables you to display and examine data from the IMS log data set in the following ways:
  - Print or copy a whole log data set
  - Print or copy from multiple log data sets based on control statement input
  - Select and print log records according to their sequential position in the data set
  - Select and print log records based upon data contained within the record itself, such as the contents of a time, date or identification field
  - Enable your exit routines to do special processing on selected log records.

See *IMS Utilities Reference: Database* for further guidance on using these utilities.

Component failures in the CICS DBCTL environment

This section discusses the impact of failures of different components of a CICS-DBCTL environment and of transaction and thread failures.

CICS failure

If CICS fails, DBCTL retains locks on database records updated by indoubt UOWs. These records remain unavailable until in-doubts are resolved. CICS records information about the disposition of UOWs on its log.

A CICS warm start or emergency restart reconstructs information describing UOWs that may be indoubt. When CICS reconnects to DBCTL, DBCTL returns a list of any indoubt UOWs. CICS notifies DBCTL of the resolution of all in-doubts, so DBCTL can commit or backout as appropriate.

If CICS fails, or if you need to cause an immediate shutdown, CICS attempts to disconnect from DBCTL. At this time, CICS gives the requests in progress time to complete before shutdown occurs. The time is specified in the DRA startup table parameter, TIMEOUT. (For information on this parameter, see "Defining the IMS DRA startup parameter table" on page 27.) If TIMEOUT is exceeded and CICS terminates while threads are still active in DBCTL, a U113 abend of DBCTL will occur. If this happens, you will have to restart DBCTL (IMS).

Choosing a value for TIMEOUT involves a trade-off between the length of restart process, which might be delayed if the value you specify is too high, and the risk of causing U113 abends, which might increase if you specify to low a value. One

possible solution is to specify a TIMEOUT value that is about equal to the average length of time between BMP checkpoints. If a BMP checkpoint has been taken, there is less likelihood that CICS resources are waiting. This lessens the likelihood of U113 abends without lengthening the restart process too much.

If you want an abnormal termination of CICS and CICS does not respond to an immediate shutdown, use an **MVS CANCEL** command. This command, and CICS abends with different causes, should not result in an IMS U113 abend because DBCTL "traps" the CANCEL and an MVS system abend code of 08E is issued instead. Changing the effect of an MVS CANCEL from a U113 abend to an MVS system abend of 08E makes the effects of a CANCEL more like the effects of a CICS immediate shutdown, as described above. If you have been obliged to cancel CICS in this way, do not start CICS with the START=INITIAL system initialization parameter unless absolutely necessary, especially if there is a possibility of indoubt units of work for DBCTL, because CICS will lose its record of the indoubt units of work.

For further information on the effects of a CICS failure in a DBCTL environment, see the section on CCTL termination in the appropriate *IMS Customization Guide: Database* .

Database resource adapter (DRA) failure

If the DRA fails:

- DBCTL notifies CICS that the DRA is terminating abnormally, and message DFHDB8106 is issued.
- CICS cleans up the storage associated with the CICS-DBCTL interface and disconnects from DBCTL.
- When it has done this, CICS issues message DFHDB8102.
- You must then reconnect DBCTL using the CDBC CONNECT command.

DBCTL failure

A termination of DBCTL should not cause CICS to terminate, it leaves CICS without DBCTL services. The DRA is left partially initialized to help reduce the restart time.

If any of the DBCTL address spaces (DBC, DBRC, or DLISAS) fails, all of these address spaces are terminated and you must restart the system using an /ERESTART command.

If you are using the IRLM as your lock manager, and it has failed as well as DBCTL, you must restart it before restarting DBCTL. See "IRLM failure" on page 80.

Normally, you terminate DBCTL with a /CHECKPOINT FREEZE or a /CHECKPOINT PURGE command, but an MVS MODIFY command can be used to force the termination of DBCTL. The STOP option used with the MODIFY command forces termination without a dump and the DUMP option forces termination with a dump. The DBCTL address space terminates with a U0020 abend. The messages received at the system console are:

```
DFS628I  ABNORMAL TERMINATION SCHEDULED DFS629I  IMS DBC REGION ABEND
jobname 0020
```

If DL/I is processing a request and the thread that is doing the processing abends is active in DL/I or is waiting on a lock, DBCTL abends with a U113 after the following message has been sent to the system console:

```
DFS613I  DBC RCN U113 DUE TO Sxxx Uyyyy DURING DL/I CALL IN CCTL
         zzzzzzzz dddd
```

where:

xxx is the system abend code. This is `S000` if it is a user abend.

yyyy is the user abend code. This is `U0000` if it is a system abend.

zzzzzzzz
is the job name of the abending CICS system or BMP.

dddd is the DBCTL system identifier.

For example, for a user abend:

```
DFS613I  DBC RCN U113 DUE TO S000 U0474 DURING DL/I CALL IN CCTL
         DBDCCICS IMSA
```

CICS is isolated from such abends because, in DBCTL, each thread TCB has its own extended subtask ABEND exit (ESTAE).

The threads are then terminated and the DRA attempts to reconnect to DBCTL. Any requests made by the subsystem during this period result in a return code of 40, which indicates that no active communications exist with DBCTL, or a return code 28, which indicates that the specified thread does not exist. These return codes are included in messages DFHDB8104, DFHDB8109, DFHDB8111, and DFHDB8130. Guidance on interpreting them is in the DBCTL DRA return codes appendix of *IMS Messages and Codes* .

The DRA attempts to reconnect to DBCTL. After the first failing attempt, you are given the opportunity to reply to message DFS690A. You can reply either WAIT, in which case the DRA continues trying to reconnect, or CANCEL, in which case the DRA stops trying to reconnect. If you reply CANCEL, you must use the CDBC transaction to reconnect DBCTL.

If you reply WAIT, the time interval between each attempt to reconnect is as specified in the DRA startup parameter TIMER (described in "Defining the IMS DRA startup parameter table" on page 27).

If you reply WAIT and later want to prevent further attempts to reconnect, use the CDBC DISCONNECT transaction. (See "Deciding whether to use orderly or immediate disconnection" on page 41.)

IRLM failure

When the IRLM fails, DBCTL subsystems using it cannot continue normal operations.

DBCTL terminates active programs that are using the IRLM with a U3303 abend and forces any PSB schedule requests to wait until it has been reconnected to the IRLM. You reconnect DBCTL to the IRLM by first restarting the IRLM using an MVS START command, and then issuing an MVS MODIFY RECONNECT command to DBCTL. For guidance on using MVS commands with the IRLM and DBCTL, see *IMS Operator's Reference* .

Transaction and thread failures

If a transaction fails in ***DBCTL***, the CICS transaction is abended.

If a transaction fails in ***CICS*** when a DL/I request it has issued is being processed in DBCTL, the error is passed to the DBCTL thread. When a transaction terminates, the thread allocated to it is released and a record is written to the IMS log. If there is an error, a return code is returned to the application in the usual form:

- For command level requests, this is to the DL/I interface block (DIB) as a status code, or transaction abend. (Definitive Programming Interface and Associated Guidance Information on what is returned to the DIB is in *IMS Application Programming: EXEC DLI Commands* .)
- For call level requests, it is to the user interface block (UIB) as a PCB status code or a transaction abend. (Definitive Programming Interface and Associated Guidance Information on what is returned to the UIB is in *IMS Application Programming: DL/I Calls* .)

  (Response codes for a DBCTL environment are in "Summary of DBCTL abends and return codes" on page 113.)

Where the transaction has been abended, the thread is also terminated, and all recoverable resources, including DL/I, are backed out. (DL/I backout is assumed on all thread and transaction failures.)

In some cases, other resources may not have been backed out, but DL/I backout has taken place. In these cases, one of the following status codes will be returned: BB, FD, FR, FS. You can also receive the FD status code on a call to a full function database if the PSB for the program (BMP) has a DEDB PCB. See "Status codes and backout" on page 97 for actions you should take if this happens.

Deadlocks and interactions with automatic restart

DBCTL detects transaction deadlocks, which can occur when two transactions are waiting for the same two resources to become available; that is, both resources are needed by both transactions.

This is described in the *CICS Recovery and Restart Guide*

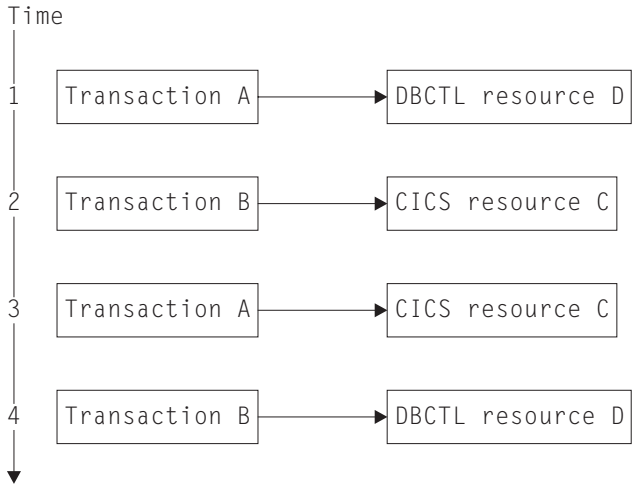


Figure 30. Transaction deadlock

If one resource is a DBCTL database and the other is a CICS resource, the task waiting for the CICS resource is abended after its DTIMOUT period has elapsed, if

you have specified one. In the example shown in Figure 30 on page 81, transaction A is the one that is abended when DTIMOUT expires. This is because it is waiting in an enqueue until transaction B frees the lock held for CICS resource C. However, CICS resource C cannot be freed because transaction B is waiting for transaction A to free the lock it is holding on DBCTL resource D.

If you did not specify DTIMOUT for the task using the CICS resource, both tasks will remain suspended indefinitely, unless one of them is canceled by the CICS master terminal operator (as described in "Purging a transaction that is using DBCTL" on page 60).

If the resources are both DBCTL databases, DBCTL detects the potential deadlock when the database requests that create the deadlocks are attempted. DBCTL then causes the task with less update activity to be abended. The abend (ADCD) causes all resources to be backed out. If a deadlock is detected when you are using DEDBs, an FD status code is issued instead of an ADCD abend. See "Status codes and backout" on page 97 for details.

For DL/I full function databases and DEDBs, if you have specified automatic restart, the task can be restarted at this point. (See *CICS Recovery and Restart Guide* for help on specifying automatic restart.) However, this can take place only if the transaction abended in the first (or only) UOW, and there has been no terminal input or output since the initial terminal input was read.

BMP failures

If a BMP fails, DBCTL backs out any changes made by that BMP following the latest successful syncpoint. You must restart BMPs, because DBCTL does not restart them automatically.

The JCL used to restart BMPs depends on whether the checkpoint for the BMP is still on an OLDS available to DBCTL. If the BMP's last checkpoint records are ***not*** in the OLDS, they will be in the SLDS, and you must add an IMSLOGR DD statement for the SLDS(s) containing the log records required to the BMP JCL. Guidance on the JCL needed to do this is in *IMS Utilities Reference: Database* .

There is an option to defer changes made to databases by backout of BMPs at emergency restart. If you specify NOBMP on the /ERESTART command, changes made to databases by BMPs are not backed out and all PSBs affected are stopped. Databases that were being updated by BMPs when the failure occurred are also stopped. You must then do batch backout for the databases that are stopped. (Batch backout will also backout the databases that were affected.) Be aware that using NOBMP may mean that the online DBCTL is restarted sooner, but it also delays data availability for the databases that were stopped by the BMP failure.

MVS, processor, or power failures

If an MVS, processor, or power failure occurs, DBRC is unable to mark the subsystem (SSYS) records in the RECON as having terminated abnormally. This means that you cannot use automatic restart. Instead, you must use the /ERESTART command with the OVERRIDE keyword to override the RECON subsystem record. Alternatively, use the DBRC command CHANGE.SUBSYS to mark the subsystem record as abnormally terminated. You will need to do this if you want to run any utilities (such as database recovery or log utilities). This is because these utilities will fail if the subsystem record is still marked as active. For information on doing this, see *IMS Utilities Reference: Database* . Backout of in-flight updates should then occur. You can then restart CICS with an AUTO (emergency)

restart. When CICS has reconnected to DBCTL, CICS decides whether any indoubt UOWs exist, and resolves them in the same way as for other failures.

Chapter 5. Application programming for DBCTL

Application programming considerations in a DBCTL environment include facilities available to application programmers with DBCTL, abends, and return codes that might be issued with DBCTL.

Programming information on DL/I requests is in *IMS Application Programming: EXEC DLI Commands* and *IMS Application Programming: DL/I Calls*.

In most cases, existing DL/I application programs should not need any changes to access databases controlled by DBCTL. See "Migrating a DL/I program to a DBCTL program" on page 105. However, consider the following:

- Your application programs will have to deal with a number of abend and response codes that may be issued with DBCTL. (See "Summary of DBCTL abends and return codes" on page 113.)
- Enhanced scheduling with DBCTL enables a PSB to be scheduled even if some of the full function databases or DEDB areas it requires are not available. (See "Enhanced scheduling" on page 94.)
- You can use the DL/I LOG request instead of the EXEC CICS WRITE JOURNALNAME command so that all DBCTL logging information is on the IMS log instead of the CICS system log. (See "LOG command and call" on page 101.)

CICS provides the following sample programs in the SDFHSAMP library to show you how to use the CALL DL/I and EXEC DLI interfaces:

Table 5. Sample programs for DL/I

Language	CALL DL/I	EXEC DLI	PSBs used
Assembler	DFH$DLAC	DFH$DLAE	DFHSAM04, DFHSAM05
COBOL	DFH0DLCC	DFH0DLCE	DFHSAM24, DFHSAM25
PL/I	DFH$DLPC	DFH$DLPE	DFHSAM14, DFHSAM15

Other product information

The information given about IMS commands is intended to help you understand the facilities available to your CICS system when you use DBCTL. The information is not part of the CICS Programming Interface and Associated Guidance Information.

Programming languages and environments for DL/I

You can write your programs in COBOL, C, PL/I, or assembler. The examples of DL/I requests in this section are in COBOL.

You have a choice of two interfaces: the **command** level interface (EXEC DLI) and the **call** level interface (using DL/I CALLs). The *IMS Application Programming: Design Guide* contains guidance on comparing the two interfaces. For programming information on the functions of EXEC DLI commands and DL/I CALLs, see *IMS Application Programming: EXEC DLI Commands* or *IMS Application Programming: DL/I Calls* s, respectively.

Issue IMS AIB call format

CICS supports IMS requests with the AIBTDLI interface and the PCB format. In addition, IMS supports application interface block (AIB) format for issuing GMSG, ICMD, and RCMD calls.

GMSG, ICMD, and RCMD calls enable DBCTL operator commands to be sent in a CICS transaction, CDBM. See "CDBM operator transaction" on page 47.

The following calls are supported:

- DELETE
- DEQUEUE
- GET UNIQUE/GET NEXT/GET NEXT IN PARENT
- GET HOLD UNIQUE/GET HOLD NEXT/GET HOLD NEXT IN PARENT
- GETMESSAGE
- ICOMMAND
- INIT
- INQY
- INSERT
- LOG
- POSITION
- RCOMMAND
- REPLACE
- ROLS
- SETS
- STAT

CICS has the following restrictions when function shipping AIB requests:

- The AIB length must be defined as 128 to 256 bytes. IMS suggests 128, but CICS enforces this range by abend code AXF7.
- Only CICS Transaction Server systems can be in a function-shipping chain if AIB requests are being issued.
- Do not specify LIST=NO on the PCB statement in the PSB if you intend to function ship AIB requests for that PCBNAME.

See *IMS Application Programming: DL/I Calls* for programming interface information about these calls, plus information about defining AIB format instead of PCB format, and on the AIBTDLI entry point for link edit.

The following table compares the AIB and PCB formats for EXEC DLI calls.

Table 6. Comparison of AIB and PCB formats for EXEC DLI calls

AIB format	PCB format
EXEC DLI GU AIB(aibname)	EXEC DLI GU USING PCB(n)
EXEC DLI GN AIB(aibname)	EXEC DLI GN USING PCB(n)
EXEC DLI GNP AIB(aibname)	EXEC DLI GNP USING PCB(n)
EXEC DLI ISRT AIB(aibname)	EXEC DLI ISRT USING PCB(n)
EXEC DLI DLET AIB(aibname)	EXEC DLI DLET USING PCB(n)
EXEC DLI REPL AIB(aibname)	EXEC DLI REPL USING PCB(n)

Table 6. Comparison of AIB and PCB formats for EXEC DLI calls (continued)

AIB format	PCB format
EXEC DLI POS AIB(aibname)	EXEC DLI POS USING PCB(n)
EXEC DLI STAT AIB(aibname)	EXEC DLI STAT USING PCB(n)
EXEC DLI QUERY AIB(aibname)	EXEC DLI QUERY USING PCB(n)
EXEC DLI DEQ AIB(aibname)	EXEC DLI DEQ[1]
EXEC DLI LOG AIB(aibname)	EXEC DLI LOG[1]
EXEC DLI REFRESH AIB(aibname)	EXEC DLI REFRESH[1]
EXEC DLI ACCEPT AIB(aibname)	EXEC DLI ACCEPT[1]
EXEC DLI SETS AIB(aibname)	EXEC DLI SETS[1]
EXEC DLI ROLS AIB(aibname)	EXEC DLI ROLS[1]
EXEC DLI GMSG AIB(aibname)	---
EXEC DLI ICMD AIB(aibname)	---
EXEC DLI RCMD AIB(aibname)	---

Note:

1. USING PCB is not required because these commands assume the IOPCB.
2. You cannot use both the AIB and the PCB in a single EXEC DLI command, but you can choose either of them for each EXEC DLI command in an application program.

For more information about these commands, see *IMS Application Programming: EXEC DLI Commands Summary*.

Enabling CICS IMS applications to use the open transaction environment (OTE) through threadsafe programming

The CICS IMS attachment facility includes a CICS IMS database adapter, DFHDBAT, that is invoked when an application program makes an IMS request. It manages the process of acquiring a thread connection into IMS, and of returning control to the application program when the IMS processing is complete.

About this task

The CICS IMS attachment facility uses the OTE to enable the CICS IMS task-related user exit (TRUE) to invoke and return from IMS without switching TCBs. In the OTE, the CICS IMS TRUE operates as a threadsafe and open API TRUE program—it is automatically enabled using the OPENAPI option on the ENABLE PROGRAM command during connection processing. This enables it to receive control on an open L8 mode TCB. Requests to IMS are also issued on the L8 TCB, so it acts as the thread TCB, and no switch to a subtask TCB is needed.

In the OTE, if the 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 IMS request, the application program runs on the CICS main TCB, the QR TCB. When it makes an IMS request and invokes the TRUE, control passes to the L8 TCB, and IMS processing is carried out. On return from IMS, 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 IMS applications decreases usage of the QR TCB, and avoids TCB switching. An ideal CICS IMS 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 IMS request, and then continues to run on the L8 TCB through any amount of IMS requests and application code, requiring no TCB. This situation produces a significant performance improvement where an application program issues multiple IMS calls. If the application program does not issue many IMS 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 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 IMS applications, you must meet the following conditions:

- CICS must be connected to IMS Version 12 or later.
- 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 IMS 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 IMS 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 GLUEs on the execution path used by the application must be coded to threadsafe standards and defined to CICS as threadsafe (for CICS IMS 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 the *CICS Application Programming Guide* 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 IMS 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 IMS 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 IMS application would occur if your program used a non-threadsafe user exit program and a non-threadsafe EXEC CICS command after every IMS request. In particular, the use of a non-threadsafe exit program on the CICS-IMS 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 IMS.

The table below shows what happens when application programs with different concurrency attributes invoke the CICS IMS TRUE when CICS is connected to different versions of IMS.

Table 7. Combinations of application programs and the CICS IMS TRUE

Program's concurrency attribute	**CICS IMS TRUE operation**	**Effect**
QUASIRENT	Threadsafe and open API	Application program runs under the CICS QR TCB. TRUE runs under an L8 TCB, and IMS requests are executed under the L8 TCB. CICS switches to and from the CICS QR TCB and the L8 TCB for each IMS request.
THREADSAFE	Threadsafe and open API	OTE exploitation. TRUE runs under an L8 TCB, and IMS 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.
REQUIRED with API(CICSAPI)	Threadsafe and open API	OTE exploitation. TRUE runs under an L8 TCB, and IMS requests are executed under the L8 TCB. The application program runs on the L8 TCB from the start. The program always uses an L8 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)	Threadsafe and open API	OTE exploitation. Not preferred for user key CICS-IMS applications (and when storage protection is active), as it causes switching from the L9 TCB to the L8 TCB and back again for every IMS request.

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

- CICS must be connected to IMS Version 12 or later.
- FORCEQR must not be set to YES.
- The CICS IMS 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 IMS 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 IMS application must use only threadsafe TRUEs or 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.

Facilities available with DBCTL

Facilities available with DBCTL include application program access to DEDBs, additional commands, calls, and keywords, increased data availability, and the ability to use BMPs.

Application program access to DEDBs

With DBCTL, your EXEC DLI and CALL DL/I application programs can access DEDBs. For an overview of the benefits of using DEDBs (including subset pointers), see “Access to data entry databases (DEDBs)” on page 8.

For programming information on using subset pointers and EXEC DL/I keywords, see *IMS Application Programming: EXEC DLI Commands* and *IMS Application Programming: DL/I Calls* s.

Command codes to manage subset pointers in DEDBs

With DEDBs, you can set and use up to eight subset pointers for each direct dependent segment type in the database description (DBD).

You must also define in the PSB, using the SENSEG statement, which subset pointers your program will use. You can then use subset pointers from within the application program together with certain command codes. “Keywords and corresponding command codes” on page 92 tells you which subset pointers you can use with which command codes.

Additional EXEC DLI keywords

You can use a number of additional EXEC DLI keywords in a CICS-DBCTL environment; they are described in the headings that follow. Each of these keywords has a corresponding CALL DL/I command code. These are shown in “Keywords and corresponding command codes” on page 92.

LOCKCLASS

The LOCKED keyword corresponds to the Q command code. You use either of these to reserve a segment so that other programs cannot update until after you have finished with it. You can associate the Q command code with a 1-character field, from A through J, but the LOCKED keyword cannot take an argument. The LOCKCLASS keyword enables you to make full use of the DEQ command.

You use the LOCKCLASS keyword, with retrieve requests only, in the same situations that the LOCKED keyword can be used. However, the LOCKCLASS keyword can take a 1-character argument, in the range B to J inclusive. You cannot use LOCKED and LOCKCLASS for the same segment.

MOVENEXT

The MOVENEXT keyword sets the subset pointer to the segment following the current segment. You can only use it with a DEDB that uses subset pointers. You can use it when retrieving, inserting, or replacing a segment. You cannot use it with a SETZERO keyword for which you have specified subset pointer values, or with the LOCKED or LOCKCLASS keywords.

MOVENEXT, which corresponds to the M command code, can take an argument, which can be a constant of up to 8 bytes or a variable of exactly 8 bytes. Each byte indicates a subset pointer and should be a single number from 1 through 8. If you use a variable that is longer than the number of subset pointers to be referenced, you should left justify the data and set the rest of the variable to blanks (for example, X'F1F3404040').

GETFIRST

The GETFIRST keyword, which corresponds to the R command code, causes the first segment in a subset to be retrieved or inserted. You can only use it when retrieving or inserting a segment in a DEDB that uses subset pointers. You can only use one GETFIRST keyword with each parent or object segment. You cannot use the GETFIRST keyword with the FIRST, LOCKED, or LOCKCLASS keywords.

GETFIRST can take a single argument, which can be a constant or a 1-byte variable. The value of the argument must be a number from 1 through 8, in character form, that indicates a subset pointer.

SET

The SET keyword, which corresponds to the S command code, causes the appropriate subset pointer to be set unconditionally to the current position, in a DEDB with subset pointers. Use the SET keyword when retrieving, inserting or replacing a segment. You cannot use it with a SETZERO keyword that has the same subset pointer value, or with the LOCKED or LOCKCLASS keywords.

SET can take an argument, which can be a constant of up to 8 bytes, or a variable of exactly 8 bytes. Each byte indicates a subset pointer and must be a single integer, in character form, from 1 through 8. If you use a variable that is longer than the number of subset pointers to be referenced, you should left justify the data and set the rest of the variable to blanks (for example, X'F1F3404040').

SETCOND

The SETCOND keyword, which corresponds to the W command code, causes the appropriate subset pointer to be set only if it is not already set to a segment. You can only use it when processing a DEDB with subset pointers. You can use

SETCOND when retrieving, inserting, or replacing a segment. You cannot use it with the SETZERO keyword that has the same subset pointer value, or with the LOCKED or LOCKCLASS keywords.

SETCOND can take an argument, which can be a constant of up to 8 bytes or a variable of exactly 8 bytes. Each byte indicates a subset pointer and must be a single number, in character form, from 1 through 8. If you use a variable that is longer than the number of subset pointers to be referenced, you should left justify the data and set the rest of the variable to blanks (for example, X'F1F3404040').

SETZERO

The SETZERO keyword, which corresponds to the Z command code, causes the appropriate segment subset pointer to be set to zero. You can only use it with DEDBs that use subset pointers. You can use SETZERO when retrieving, inserting, replacing, or deleting a segment. You cannot use it with SET, SETCOND, or MOVENEXT keywords that have the same subset pointer values. You cannot use it with the LOCKED or LOCKCLASS keywords.

SETZERO can take an argument, which can be a constant of up to 8 bytes or a variable of exactly 8 bytes. Each byte indicates a subset pointer and must be a single number, in character form, from 1 through 8. If you use a variable that is longer than the number of subset pointers to be referenced, you should left justify the data, and set the rest of the variable to blanks (for example, X'F1F3404040').

System service (SYSSERVE)

If your application program issues a system service request in an EXEC DLI environment, you do not need to specify the PCB number, because the IOPCB is assumed for this type of request. However, if you are using one of the following EXEC DLI system service requests:

- LOG command
- REFRESH command
- ACCEPT command
- SETS command
- ROLS command (without the USING PCB(1) option)

first issue a PSB schedule command specifying the SYSSERVE keyword. See “PSB schedule command and call” on page 100 for the format of the schedule request.

Keywords and corresponding command codes

Table 8 lists EXEC DLI keywords and corresponding DL/I CALL command codes that are valid in a DBCTL environment.

Table 8. Keywords and corresponding command codes

EXEC DLI keyword	DL/I CALL command code	Purpose
KEYS	C	Using the concatenated key of a segment to identify the segment.
INTO or FROM specified on segment level to be retrieved or inserted	D	Retrieving or inserting a sequence of segments in a hierarchic path using only one request, instead of having to use a separate request for each segment (path call or command).
FIRST	F	Backing up to the first occurrence of a segment under its parent when searching for a particular segment occurrence. Disregarded for a root segment.

Table 8. Keywords and corresponding command codes (continued)

EXEC DLI keyword	DL/I CALL command code	Purpose
LAST	L	Retrieving the last occurrence of a segment under its parent.
MOVENEXT **1**	M **1**	Moving a subset pointer to the next segment occurrence after your current position.
Leaving out the SEGMENT option for segments you do not want replaced	N	Designating segments you do not want replaced, when replacing segments after a get hold request. Used when replacing part of a path of segments.
SETPARENT	P	Setting parentage at a higher level than usual. (It is usually the lowest SSA level of the call.)
LOCKED **2** LOCKCLASS **2**	Q **2**	Reserving a segment so that other programs will not be able to update it until after you have finished processing and updating it.
GETFIRST **1**	R **1**	Starting search with the first segment occurrence in a subset.
SET **1**	S **1**	Unconditionally setting a subset pointer to the current position.
No EXEC equivalent	U	Limiting the search for a segment to the dependents of the segment occurrence on which position is established.
CURRENT	V	Using the current position at this hierarchic level and above as qualification for the segment.
SETCOND **1**	W **1**	Setting a subset pointer to your current position, if the subset pointer is not already set.
SETZERO **1**	Z **1**	Setting a subset pointer to zero.

Note:

1. DEDB subset pointer operations only. These command codes are new for CICS users who are new to DBCTL.
2. Cannot be used with DEDBs.

POS command and call

With DEDBs, you can use the position (POS) command and call to retrieve the location of a specific sequential dependent segment or the location of the last inserted sequential dependent segment. The POS command and call also provides information about unused space.

You can specify only one SSA with the POS request; that is, either the root segment, or a sequential dependent segment. You can use POS to locate a specific sequential dependent segment when you already have a valid position of a root segment. If you do not already have one, you must first issue a separate POS request, or other request, to establish the position of a root segment.

The format of the POS ***command*** is:

```
EXEC DLI POS|POSITION
          USING PCB(n)
          INTO(data-area)
          [KEYFEEDBACK(area)[FEEDBACKLEN(expression)]]
          [SEGMENT(name)|SEGMENT((area))]
          [WHERE(qualification_statement)[FIELDLENGTH(expression)]]
```

Figure 31. EXEC DLI POS command

The format of the POS ***call*** is:

```
CALL 'CBLTDLI' USING POS,dedb_pcb,i/o_area[,ssa]
```

See "Keywords and corresponding command codes" on page 92 and "Comparing EXEC DLI commands and DL/I calls" on page 103 for brief comparisons of commands and calls. For further guidance on the differences between commands and calls, see *IMS Application Programming: Design Guide*.

Addressing and residency mode

Addressing mode (**AMODE**) refers to the address length that a program is prepared to handle: 24-bit addresses, 31-bit addresses, or both (ANY). Programs with an addressing mode of ANY must have been designed to receive control in either 24- or 31-bit addressing mode.

Residency mode (**RMODE**) specifies where a program is expected to reside in virtual storage. RMODE 24 indicates that a program is coded to reside in virtual storage below 16MB. RMODE ANY indicates that a program is coded to reside anywhere in virtual storage.

See the *z/OS MVS Programming: Extended Addressability Guide* for more information on AMODE and RMODE. See also the appropriate programming guides for COBOL and PL/I for guidance on placing parameters above or below the line.

With remote DL/I and DBCTL, programs can be AMODE(31) RMODE(any) with parameters above the 16 MB line, for both DL/I call and command level.

Enhanced scheduling

DBCTL supports enhanced scheduling. That is, PSB scheduling completes successfully, even if some of the full function databases or DEDB areas it requires are not available.

Full function databases that have been stopped or locked by the commands /STOP, /DBRECOVERY, or /LOCK, or that are unavailable for update because a /DBDUMP command has been issued, do not cause scheduling failures. Instead, the application program is prevented from accessing only the unavailable database(s) or area(s). Application programs can have read access to databases that have been made unavailable for update by the /DBDUMP command. If a program issues a call to an unavailable database or area, a transaction abend is issued. To avoid this happening, you can issue requests, after a PSB has been scheduled, to obtain information regarding the availability of each database and to indicate that your program will handle data availability status codes. These requests are described in "Obtaining information about database availability" and "Accepting database availability status codes" on page 96.

Obtaining information about database availability

A PSB scheduling request places data availability status codes in each of the DB PCBs.

About this task

You can use DL/I requests to obtain and refresh this information, as described below.

QUERY and REFRESH DBQUERY commands

In a command-level environment, issue the following command after a PSB schedule request for each PCB:

```
EXEC DLI QUERY PCB(n)
```

where n is the number of a PCB.

This obtains the status code and other information in the DL/I interface block (DIB). You should get one of the following values in the DIB:

- TH, which means that a PSB has not yet been scheduled and results in a DHTH abend.
- NA, which means that at least one of the databases that can be accessed using this PCB is unavailable, but does not result in an abend.
- NU, which means that at least one of the databases that can be updated using this PCB is unavailable and does not result in an abend.
- (blanks), mean that the data accessible using this PCB is available for all functions that the PCB sensitivity allows.

DIBDBORG, which is returned when DIBSTAT has been set to NA, NU or `bb` (blanks). DIBDBORG contains one of the following values describing the database organization:

- DEDB
- GSAM
- HDAM
- HIDAM
- HISAM
- INDEX
- HSAM
- SHISAM
- SHSAM.

DIBDBDNM, which is returned when DIBSTAT has been set to NA, NU or blanks, and contains the DBDNAME. You can refresh these status codes using the command:

```
EXEC DLI REFRESH DBQUERY
```

INIT call: format for refreshing status code information

Application programs using the DL/I CALL interface can access the PCB status codes directly.

You can refresh these status codes using the INIT call as follows:

```
CALL 'CBLTDLI' USING INIT,i/o pcb,i/o_area
```

where `i/o_area` contains a string in the format `LLZZcharacter_string`.

- `LL` is a halfword containing the length of the `character_string` including `LLZZ`.
- `ZZ` contains binary zeros
- `character_string` contains DBQUERY.

The data availability status codes used in this context are:

- (blanks), which means that all of the databases are available.
- NA, which means that at least one of the databases that can be ***accessed*** using this PCB is unavailable.
- NU, which means that at least one of the databases that can be ***updated*** using this PCB is unavailable for update.

Accepting database availability status codes

You can use DL/I requests to indicate that your application program is prepared to accept and handle database availability status codes for DL/I calls.

This is described in "ACCEPT STATUSGROUP command" and "INIT call: format for accepting status codes." These status codes may have been issued because PSB scheduling has completed without all of the referenced databases being available.

ACCEPT STATUSGROUP command

For ***command*** level application programs, use:

```
EXEC DLI ACCEPT STATUSGROUP('A')
```

INIT call: format for accepting status codes

For ***call*** level application programs, use:

```
CALL 'CBLTDLI' USING INIT,i/o pcb,i/o_area
```

where `i/o_area` contains a string in the format `LLZZcharacter_string`.

- `LL` is a halfword containing the length of the `character_string` including `LLZZ`
- `ZZ` contains binary zeros
- `Character_string` contains STATUSGROUPA.

If you have used ACCEPT STATUSGROUP, and a DL/I request tries to access a database or a DEDB area that is not available after PSB schedule, DBCTL returns a status code instead of abending the transaction. If you have not used ACCEPT STATUSGROUP, the transaction will be abnormally terminated with ADCI if it tries to access unavailable data. (See "Summary of DBCTL abends and return codes" on page 113 for details of accompanying return codes.)

The status codes used are:

- (blanks), which means that the request completed successfully.
- BA, which means that the request could not be completed because a database was not available. In this case, only the updates done for the current DL/I call are backed out.
- BB, which means that the request could not be completed because a database was not available. In this case, all DL/I updates are backed out to the last commit point.
- BC, which means that the request could not be completed because of a deadlock.

Note: Only DL/I resources are backed out because the transaction has not abended. Therefore, ensure that you keep DL/I and other resources in synchronization.

See *IMS Application Programming: EXEC DLI Commands* or *IMS Application Programming: DL/I Calls* s for programming information on status codes.

Although a PSB can contain PCBs for GSAM and MSDB databases, and the PSB can be scheduled, programs using DBCTL (or any other kind of CICS-DL/I program) cannot access those GSAM or MSDB databases online from CICS. Access to such databases is by means of batch and BMPs only. See "I/O PCB" on page 98 for information on the option SCHD, which you can use to state whether you require an input/output PCB (I/O PCB).

Status codes and backout

The following DEDB status codes are returned when DL/I backout has taken place: BB, FD, FR, FS.

If you receive one of these status codes, it is as if any update requests you issued to full function databases or to DEDBs in the same UOW had not taken place.

If you are using EXEC DLI, these status codes are, as usual, accompanied by a DHBB, DHFD, DHFR, or DHFS abend.

If you are using CALL DL/I and if you want any other resources you may have been updating in the same UOW to be backed out, issue an **EXEC CICS ABEND** request or a SYNCPOINT ROLLBACK command.

Batch message processing programs (BMPs)

Batch message processing programs (BMPs) are application programs that perform batch type processing online and can access databases controlled by DBCTL.

You can run the same program as a BMP or as a batch program. Figure 32 on page 98 shows the kind of data BMPs can access. See *IMS Application Programming: Design Guide* for further guidance on using BMPs.

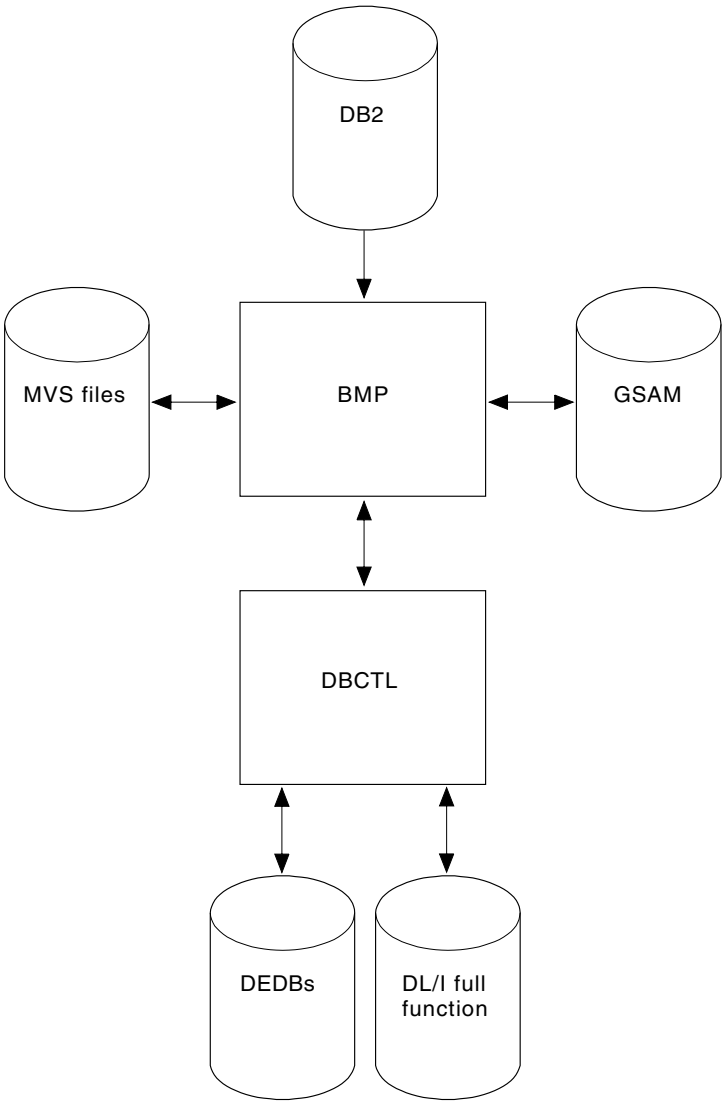


Figure 32. BMP access

System service requests

I/O PCB

A PSB used in a DBCTL environment can contain any of these PCB types.

- **I/O PCB**. In a CICS-DBCTL environment, an input/output PCB (I/O PCB) is needed to issue DBCTL service requests. Unlike other types of PCB, it is not defined with PSB generation. If the application program is using an I/O PCB, this has to be indicated in the PSB scheduling request, as explained in "Format of a PSB" on page 99.
- **Alternate TP PCB(s)**. An alternate TP PCB defines a logical terminal and can be used instead of the I/O PCB when it is necessary to direct a response to a terminal. Alternate TP PCBs appear in PSBs used in a CICS-DBCTL environment, but are used only in an IMS/VS DC or IMS TM environment. CICS applications using DBCTL cannot successfully issue requests that specify an alternate TP PCB, an MSDB PCB, or a GSAM PCB, but PSBs that contain this kind of PCB can be scheduled successfully in a CICS-DBCTL environment.

Alternate PCBs are included in the PCB address list returned to a call level application program. The existence of alternate PCBs in the PSB can affect the PCB number used in the PCB keyword in an EXEC DLI application program, depending on whether you are using CICS online, batch programs, or BMPs. For more information, see "PCB summary" below.

- **DB PCB(s)**. A database PCB (DB PCB) is the PCB that defines an application program's interface to a database. One DB PCB is needed for each database view used by the application program. It can be a full function PCB, or a DEDB PCB.
- **GSAM PCB(s)**. A GSAM PCB defines an application program's interface for GSAM operations.

With DBCTL, a CICS online application program receives, by default, a DB PCB as the first PCB in the parameter list passed to it after scheduling.

With the EXEC DLI interface, in order to use system service requests, you specify the SYSSERVE keyword on the SCHD command to indicate that your application program can handle an I/O PCB. In an EXEC DLI environment, the SYSSERVE keyword does not change the PCB numbering, which means that your first PCB is still the DB PCB, and you do not need to specify a PCB number when you issue a system service request.

With the DL/I CALL interface, in order to use system service requests, you use the IOPCB parameter on the PCB to indicate that your application program can handle an I/O PCB. The I/O PCB will then be the first PCB in the parameter address list passed back to your application program.

Format of a PSB

PSBs used in a DBCTL environment will be of the following form:

```
[IOPCB]
[Alternate TP PCB ... Alternate TP PCB]
[DBPCB ... DBPCB]
[GSAMPCB ... GSAMPCB]
```

Figure 33. General format of a PSB in a DBCTL environment

Each PSB must contain at least one PCB. A DB PCB can be a full function PCB, or a DEDB PCB.

PCB summary

This section summarizes information concerning I/O PCBs and alternate PCBs in the supported environments.

Read it if you intend to issue system service requests.

CICS online programs:

- EXEC DLI

  The first PCB in your PCB address list always refers to the first database PCB (DB PCB) whether or not you specify the SYSSERVE keyword.
- CALL DL/I

  If you specify the IOPCB option on the PCB call, the first PCB in your PCB address list will be the I/O PCB, followed by any alternate PCBs, followed by the DB PCBs.

  If you do not specify the IOPCB option, the first PCB in your PCB address list will be the first DB PCB.

BMPs:

- EXEC DLI and CALL DL/I

  The PCB list always contains the address of the I/O PCB, followed by the addresses of any alternate PCBs, followed by the addresses of the DB PCBs.

Batch programs:
Alternate PCBs are always returned to batch programs irrespective of whether you have specified CMPAT=Y. The I/O PCB is returned depending on the CMPAT option, as follows:

- EXEC DLI and CALL DL/I

  If you specify CMPAT=Y, the PCB list contains the address of the I/O PCB, followed by any alternate PCBs, and then the DB PCBs.

  If you do not specify CMPAT=Y, the PCB list contains the addresses of any alternate PCBs followed by the addresses of the DB PCBs.

Table 9 summarizes the I/O PCB and alternate PCB information.

Table 9. PCB summary

Environment	EXEC DLI: I/O PCB count included in PCB(n)	EXEC DLI: Alternate PCB count included in PCB(n)	CALL DLI: I/O PCB address returned	CALL DLI: Alternate PCB address returned
CICS DBCTL: SCHD request issued without the IOPCB or SYSSERVE option	No	No	No	No
CICS DBCTL: SCHD request issued with the IOPCB or SYSSERVE for a CICS DBCTL request or for a function shipped request which is satisfied by a CICS system using DBCTL	No	No	Yes	Yes
BMP	Yes	Yes	Yes	Yes
Batch: CMPAT=N specified	No	Yes	No	Yes
Batch: CMPAT=Y specified	Yes	Yes	Yes	Yes

PSB schedule command and call

The format of the schedule ***command*** is:

```
EXEC DLI SCHD PSB(name)[SYSSERVE]
```

Specifying SYSSERVE does not affect the PCB number you specify in the USING PCB keyword because PCB(1) will always refer to the first DB PCB. The application program must establish addressability to the I/O PCB. See *IMS Application Programming: Design Guide* for further guidance on doing this.

The format of the schedule ***call*** is:

```
CALL 'CBLTDLI' USING PCBb,psbname,uibptr[,sysserve]
```

where sysserve is an optional 8-byte variable, set to either IOPCB or NOIOPCB.

Almost all the new DL/I calls supported in the CICS-DBCTL environment require an I/O PCB. The two exceptions are the ROLS call, which can use a DB PCB, and the POS call, which uses a DEDB PCB.

Preventing DHxx abends after EXEC DLI SCHD PSB failure:

When a PSB schedule request fails (for example, because a database is unavailable), CICS abends the transaction with a DHxx abend code.

In a production system, PSB schedule request failures are more likely to be caused by unavailability of a database than by application coding errors, which means that end users may see DHxx abends unnecessarily. To prevent this happening, you can use the EXEC DLI SCHD PSB keyword, NODHABEND, which specifies that no DHxx abends are issued for that PSB schedule request. Instead, the xx value is returned to the application program in DIBSTAT, enabling the application to deal with the situation in a more user-friendly way, and avoiding the need to code global HANDLE ABENDs (EXEC DLI does not support HANDLE CONDITION).

DEQ command and call

The DEQ (dequeue) request releases segments that were retrieved using the LOCKCLASS keyword or the Q command code.

The LOCKED keyword cannot take an argument, and cannot be used with DEQ. (Segments locked using the LOCKED keyword are released when a SYNCPOINT is taken.) Instead, you use LOCKCLASS with DEQ, which can take a 1-character argument in the range B to J inclusive. (These keywords correspond to the Q command code, which you can associate with a 1-character field in the range A to J.) You cannot use LOCKED and LOCKCLASS for the same segment. Using LOCKCLASS or Q on retrieval requests enables you to reserve segments for exclusive use by your transaction. No other transaction is allowed to update these reserved segments until your transaction reaches a syncpoint, or the DEQ request has been issued, when the reserved segments are released. This means that your application can leave these segments and retrieve them later without them being changed in the meantime.

The format of the DEQ ***command*** is:

```
EXEC DLI DEQ LOCKCLASS(data_value)
```

where data_value is a 1-byte alphabetic character in the range B to J.

The format of the DEQ ***call*** is:

```
CALL 'CBLTDLI' USING function,i/o pcb,i/o_area
```

where function is the address of a 4-byte area that contains the value of the DEQb function, i/o pcb is the name of the I/O PCB (mandatory), and i/o_area is a 1-byte alphabetic character in the range A to J.

LOG command and call

You can use the LOG request ***online*** when you want a record to be written from an application program to the IMS log.

Your program can specify whatever information you want to be on the log. You may prefer to use it instead of EXEC CICS journal commands so that all your DBCTL information will be on the IMS log instead of the CICS log. IMS uses different log codes to distinguish different types of log record. All user log records in the IMS log have the same code. Records logged using the LOG request will not be backed out if synchronization fails and the UOW is canceled.

The format of the LOG ***command*** is:

```
EXEC DLI LOG FROM(area) LENGTH(expression)
```

The format of the LOG ***call*** is:

```
CALL 'CBLTDLI' USING LOGb,i/o-pcb,data-area
```

where LOGb is the address of a 4-byte area that contains the value of the LOGb function.

Defining intermediate backout points for DBCTL resources

About this task

The SETS and ROLS requests enable you to define multiple points at which to preserve the state of DL/I full function databases and to return to these points later. The backout points are ***not*** CICS syncpoints, they are intermediate backout points that apply only to DBCTL resources. For example, you can use them to allow your program to handle the consequences of PSB scheduling having completed without all of the referenced DL/I databases being available.

The SETS and ROLS requests apply to DL/I full function databases only. If an UOW is updating recoverable resources other than full function databases, for example, DEDBs and VSAM files, the SETS and ROLS requests have no effect on the non-DL/I resources. Therefore, take steps to ensure the consistency of other resources involved, if any. See "Summary of DBCTL abends and return codes" on page 113 for explanations of relevant return codes.

SETS command and call:
You can use a SETS request to define points in your application at which to preserve the state of DL/I databases before initiating a set of DL/I calls to perform a function. Your application can issue a ROLS request later if it cannot complete that function.

The format of the SETS ***command*** is:

```
EXEC DLI SETS [TOKEN(mytoken) AREA(data-area)]
```

where mytoken is a 4-byte token associated with the current processing point.

data-area is an area to be restored to the program when a ROLS request is issued. The first two bytes of the data-area field contain the length of the data-area, including the length itself. The second two bytes must be set to X'0000'.

The format of the SETS ***call*** is:

```
CALL 'CBLTDLI' USING SETS,i/o_pcb[,i/o_area,token]
```

TOKEN(mytoken) AREA(data-area) in the command version and i/o_area,token in the call version are optional, but if you do omit them, this cancels any intermediate backout points set in previous SETS requests and ROLS backs out to the last commit point.

ROLS command and call:
You can use the ROLS request to backout to the state all full function databases were in before: (a) a specific SETS request or (b) the most recent commit point.

The format of the ROLS ***command*** is:

```
EXEC DLI ROLS [TOKEN(mytoken) AREA(data-area)]
```

The format of the ROLS ***call*** is:

```
CALL 'CBLTDLI' USING ROLS,pcb[,i/o_area,token]
```

i/o_area and token on the call, and TOKEN(mytoken) AREA(data-area) on the command are optional. If you include them, ROLS backs out to the SETS ***you specified***. If you omit them, ROLS backs out to the ***most recent*** SETS.

The ROLS ***command*** has a second format, the purpose of which is to backout to ***before*** an ACCEPT STATUSGROUPA request:

```
EXEC DLI ROLS [USING(PCB(n)]
```

where n is the name of a database PCB that has received a "data" unavailable status code. This causes the same action to take place that would have occurred had the program not issued an ACCEPT STATUSGROUPA request. (See "Accepting database availability status codes" on page 96.)

Comparing EXEC DLI commands and DL/I calls

Use the following table to compare corresponding EXEC DLI and CALL DL/I requests and their functions.

These commands and calls are threadsafe.

Table 10. EXEC commands and DL/I calls

EXEC DLI	CALL DL/I	Function
GU, GN, and GNP	GU, GN, and GNP	Retrieving segments from the database
GU, GN, and GNP	GHU, GHN, and GHNP	Retrieving segments from database for updating
DLET	DLET	Deleting segments from a database
REPL	REPL	Replacing segments in a database
ISRT	ISRT	Adding segments to a database
LOAD	ISRT	Initially loading a database
SCHD	PCB	Scheduling a PSB
TERM	TERM	Terminating a PSB
CHKP	CHKP (basic)	Issuing a basic checkpoint
SYMCHKP	CHKP (extended)	Issuing a symbolic checkpoint
XRST RETRIEVE	XRST	Issuing an extended restart
-----[1]	SYNC	Requesting sync point processing
DEQ	DEQ	Releasing segments retrieved using Q command code
-----[1]	GSCD	Retrieving system addresses
LOG	LOG	Writing a message to the system log
ROLL or ROLB	ROLL or ROLB	Dynamically backing out changes
STAT	STAT	Obtaining system and buffer pool statistics (see also Table 11 on page 104)
REFRESH ACCEPT QUERY[2]	INIT	Refreshing, accepting, and querying data availability status codes

Table 10. EXEC commands and DL/I calls (continued)

EXEC DLI	CALL DL/I	Function
SETS	SETS	Setting a backout point
ROLS	ROLS	Backing out to a previously set backout point
-----[1]	GSAM	Issuing requests to GSAM databases
POS	POS	Retrieving positioning or space usage information in a DEDB area

Note:

1. No EXEC DLI equivalent. Use a DL/I CALL, but note that you cannot mix EXEC and CALL in the same UOW.
2. Status codes are available directly to CALL DL/I applications. EXEC DLI QUERY corresponds to code in the CALL DL/I program instructing it to examine the PCB.

DL/I requests supported

This table summarizes the DL/I requests you can use and the environments in which they apply.

Table 11. DL/I requests supported

Request type	CICS and DBCTL[1]	Batch	BMP
Get commands and calls (GU, GHU, GN, GHN, GNP, GHNP)	Yes	Yes	Yes
DLET command and call	Yes	Yes	Yes
REPL command and call	Yes	Yes	Yes
ISRT command and call	Yes	Yes	Yes
ISRT call (initial load)	No	Yes	No
LOAD command	No	Yes	No
PCB call	Yes	No	No
SCHD command	Yes	No	No
TERM command and call	Yes	No	No
CHKP command and call (basic)	No	Yes	Yes
CHKP call (extended)	No	Yes	Yes
SYMCHKP command	No	Yes	Yes
XRST command and call	No	Yes	Yes
RETRIEVE command	No	Yes	Yes
SYNC call	No	No	Yes
DEQ command and call	Yes	Yes	Yes
GSCD call	No	Yes	No
LOG call	Yes	Yes	Yes
LOG command	Yes	Yes	Yes
ROLL call	No	Yes	Yes
ROLL command	No	Yes	Yes
ROLB command and call	No	Yes	Yes
STAT command and call	Yes[2]	Yes[2]	Yes[2]

Table 11. DL/I requests supported (continued)

Request type	CICS and DBCTL[1]	Batch	BMP
INIT call	Yes	Yes	Yes
REFRESH command	Yes	Yes	Yes
ACCEPT command	Yes	Yes	Yes
QUERY command	Yes	Yes	Yes
SETS command and call	Yes	Yes	Yes
ROLS command and call	Yes	Yes	Yes
GSAM calls	No	Yes	Yes
POS command and call	Yes	No	Yes

Note:

1. Requests are also supported with ***function shipping*** to a remote CICS that uses ***DBCTL***.
2. For programming information on keywords used to request the enhanced statistics, see *IMS Application Programming: DL/I Calls* .

Migrating programs to DBCTL

Read this section if the version of CICS that you are migrating from is earlier than CICS Transaction Server, Version 1.1.

Considerations for migrating programs to DBCTL include using your existing local DL/I programs with DBCTL, and changing CICS shared data base programs and "native" IMS batch jobs to run as BMPs.

Migrating a DL/I program to a DBCTL program

About this task

Your existing CICS application programs should not require any changes in order to run in the DBCTL environment.

However, you must define the names of all DMBs to be owned by DBCTL to DBCTL using system definition DATABASE statements. Make sure that you have defined the names of all PSBs to be used by application programs when accessing DBCTL databases using system definition APPLCTN statements (which are equivalent to DFHDLPSB in local DL/I). All DMBs to be owned by a given PSB must be owned by the same DBCTL. See *IMS Application Programming: Design Guide* for further guidance on defining PSBs.

Your applications may receive some different abend codes. You may also get a message that DL/I is not available. This may occur because DBCTL can be disconnected dynamically from CICS, using the CDBC transaction, and because, unlike local DL/I, a failure in DBCTL should not cause CICS to fail, but merely leaves it without DL/I services. New abend codes are summarized in "Summary of DBCTL abends and return codes" on page 113.

An application program that updates DL/I databases owned by DBCTL and has activated an exit to use HANDLE ABENDs, should terminate the abend exit routine with an ABEND request.

We recommend that programs that have read-only access to the database, and an abend exit is active, should not attempt to reschedule a PSB as part of abend processing. This is because if the high order bit of the DBCTL return code (PAPLRETC) is set on, the DBCTL thread has been withdrawn from use by the transaction and any further DBCTL request is abended with a code of AEY9. The only exception to this is if the abending request was a schedule, this is because the thread is not obtained until the schedule completes successfully.

Migrating CICS shared database batch jobs to BMPs

With CICS Transaction Server for z/OS, Version 4 Release 2, you must migrate any batch jobs that currently use the CICS shared database facility to BMPs so that they communicate directly with the DBCTL address space.

About this task

BMPs perform batch processing and are started with job control language (JCL) like programs in a batch environment. The JCL for this is in the IMS procedure IMSBATCH. (For further guidance on IMSBATCH, see *IMS System Definition Reference* or *IMS Installation Volume 2: System Definition and Tailoring*.) Migrating these batch jobs gives you:

- A performance advantage, because BMPs communicate directly with DBCTL instead of accessing databases through CICS. For more information on BMP performance, see Chapter 8, “Statistics, monitoring, and performance for DBCTL,” on page 141.
- The ability to use system service requests, such as symbolic checkpoint (CHKP) and extended restart (XRST).
- Access to DEDBs.
- Access to GSAM databases.
- Logging to the IMS log (so there is no need for multiple logs).
- Automatic restart from last checkpoint without requiring JCL changes.

Automatic backout, which you will already be using for your shared database programs, also applies to BMPs.

With BMPs, the PCB always includes an I/O PCB. If you have specified CMPAT=Y in the JCL to execute your CICS shared database job, need not change any source code in your application. If you have specified CMPAT=N, change your code to allow for the addition of an I/O PCB. For example, in COBOL, you do this by changing the ENTRY statement in the PROCEDURE division to include the I/O PCB. For guidance on doing this, in COBOL, PL/I and assembler language, see *IMS Application Programming: EXEC DLI Commands* or *IMS Application Programming: DL/I Calls* s.

Migrating native IMS batch jobs to BMPs

You are advised to migrate “native” IMS batch jobs to BMPs that use DBCTL.

About this task

This will give you:

- Logging to the IMS log (no need for multiple logs).
- Automatic restart from the last checkpoint (no JCL changes required).
- Concurrent access to databases.

- Automatic backout. (You may already have this for your batch programs if you use disk logging.)

General design considerations for BMPs

Your applications must take checkpoints and must be restartable from the last checkpoint (also known as checkpoint restart). This is particularly important for batch programs migrated to BMPs. A disconnection request cannot complete until a BMP checkpoint occurs if a CICS thread is waiting for a lock held by a BMP.

Design and code your batch programs to be restartable from checkpoints, even if you have no immediate intention of running them as BMPs. This is because it is simpler to design batch programs with checkpoint restart than to introduce it to existing programs if you do decide to migrate them later.

The following is a summary of what to consider when designing BMPs and applications to run in a DBCTL environment:

- All BMPs and applications should issue frequent checkpoints to avoid locking out other resource users.
- All BMPs and applications must be restartable from last checkpoint. This is because records in the same database may have since been updated, and these updates would be lost if the database were restored from a previous backup.
- BMPs and applications should not hold on to locks for long periods without issuing checkpoints or syncpoints (either explicitly or implicitly).
- Beware of long-running applications that do not issue syncpoints or that hold data over terminal conversations.
- Be aware that small but very frequently updated databases may cause contention for resources.
- Review the use of control records; that is, records that are accessed by most applications. If they have to be updated, it is important to remember that the CI or physical block is locked from other subsystems until the updates are committed.

Using global user exit XDLIPRE to change PSB to be scheduled

You can use the global user exit XDLIPRE to change the PSB name that the application program has scheduled at execution time.

This section contains Product-sensitive Programming Interface information.

Figure 34 on page 108 contains an example of XDLIPRE that you can copy and modify. Note that this example is provided for guidance only. See the *CICS Customization Guide* for programming information on global user exits.

You can also use the XDLIPRE exit to change the identity of the SYSID, enabling work to be rerouted from a SYSID that becomes unavailable to one that is available.

```
*********************************************************************
* This is an example for global user exit XDLIPRE                   *
*                                                                   *
* It is invoked before any DLI call being passed to              *
* the remote or DBCTL processors.                             *
*                                                                   *
* A check is made for the presence of a PSB.                        *
* If not, a normal return is made                                   *
*                                                                   *
* If the PSB is in a predefined table, it is changed to a           *
* different value, and a normal return is made.                     *
*                                                                   *
* If not, set PSB name to blanks and normal return.                 *
*                                                                   *
* In all cases,a trace entry is written describing the action       *
* taken, using TRACE-POINT 384  (hex '0180')                        *
*                                                                   *
*********************************************************************
*                                                                   *
* The first few instructions set up the global user exit            *
* environment, identify the user exit point, prepare for the use of*
* the exit programming interface, and copy in the definitions that *
* are to be used by the XPI function.                               *
*                                                                   *
*********************************************************************
*
         DFHUEXIT TYPE=EP,ID=XDLIPRE    PROVIDE DFHUEPAR PARAMETER
*                                       LIST AND LIST OF EXITID
*                                       EQUATES
*
         DFHUEXIT TYPE=XPIENV           SET UP ENVIRONMENT FOR
*                                       EXIT PROGRAMMING INTERFACE
*                                       MUST BE ISSUED BEFORE ANY
*                                       XPI MACROS ARE ISSUED
```

Figure 34. Example of XDLIPRE user exit to change PSB names 1/6

```
*
         COPY  DFHTRPTY                DEFINE PARAMETER LIST FOR
*                                      USE BY DFHTRPTX MACRO
*
         COPY  DFHSMMCY                DEFINE PARAMETER LIST FOR
*                                      USE BY DFHSMMCX MACRO
*
*****************************************************************
*The following DSECT maps a storage area to be used as work area *
*for the information in the TRACE entry.                         *
*****************************************************************
*
DSA      DSECT                         DSECT FOR GETMAINED STORAGE
         USING DSA,R7
*
RETCODE  DS    F                       store return code
MESSAGEA DS    F                       message address for trace
MESSAGEL DS    F                       message length for trace
MESSAGE  DS    0CL37
OLDPSB   DS    CL8
MESS1    DS    CL21
NEWPSB   DS    CL8
*********************************************************************
*The next  instructions  form  the  normal  start  of a global user *
*exit program, setting the program addressing mode to 31-bit, saving*
*the calling program's registers, establishing base addressing*
*and establishing the addressing of the user exit parameter list.   *
*********************************************************************
*
DLIPR    CSECT
DLIPR    AMODE 31
*
         SAVE (14,12)                  SAVE CALLING PROGRAM'S RGSTRS
*
         LR    R11,R15                 SET UP USER EXIT PROGRAM'S
         USING DLIPR,R11               BASE REGISTER
*
         LR    R2,R1                   SET UP ADDRESSING FOR USER
         USING DFHUEPAR,R2             EXIT PARAMETER LIST -- USE
*                                      REGISTER 2 AS XPI CALLS USE
*                                      REGISTER 1
*
*********************************************************************
*Before issuing an XPI function call, set up addressing to XPI     *
*parameter list.                                                   *
*********************************************************************
*
         L     R5,UEPXSTOR             SET UP ADDRESSING FOR XPI
*                                      PARAMETER LIST
```

Figure 35. Example of XDLIPRE user exit to change PSB names 2/6

```
*********************************************************************
*  Before issuing an XPI function call, you must ensure that register*
*  13 addresses the kernel stack.                                   *
*********************************************************************
*
         L     R13,UEPSTACK             ADDRESS KERNEL STACK
*
*********************************************************************
*  Issue a GETMAIN to get storage for work area                     *
*********************************************************************
*
         USING DFHSMMC_ARG,R5           MAP PARAMETER LIST
*
         DFHSMMCX CALL,                                              X
               CLEAR,                                                X
               IN,                                                   X
               FUNCTION(GETMAIN),                                    X
               GET_LENGTH(100),                                      X
               STORAGE_CLASS(USER),                                  X
               SUSPEND(NO),                                          X
               OUT,                                                  X
               ADDRESS((R7)),                                        X
               RESPONSE(*),                                          X
               REASON(*)
*
*********************************************************************
*  SET UP THE NORMAL RETURN CODE                                    *
*********************************************************************
*
         LA    R6,UERCNORM
         ST    R6,RETCODE
*
*********************************************************************
*  See if a PSB exists                                              *
*********************************************************************
*
         L     R6,UEPPSBNX              PSB EXISTENCE FLAG
         TM    0(R6),UEPPSB1            PSB EXISTS?
         BO    PSBCALL                  YES
         MVC   MESSAGE,MESS3T           NO-MOVE MESSAGE TO DSA
         B     TRACE
*
*********************************************************************
*  See if we want to change a PSB name                              *
*********************************************************************
*
PSBCALL  EQU   *
         L     R6,UEPPSBNM              ADDRESS OF PASSED PSB NAME
         LA    R8,PSBS                  ADDRESS OF table of PSB pairs
         CLC   0(8,R6),0(R8)            SAME?
```

Figure 36. Example of XDLIPRE user exit to change PSB names 3/6

```
         BE    FOUND                   YES
         LA    R8,16(R8)               BUMP TO NEXT PAIR
         CLC   0(8,R6),0(R8)
         BE    FOUND
         LA    R8,16(R8)               BUMP TO NEXT PAIR
         CLC   0(8,R6),0(R8)
         BE    FOUND
         B     NOTFOUND                NO MATCH - END
*
*********************************************************************
*  Move new PSB name in                                             *
*********************************************************************
*
FOUND    EQU   *
         MVC   0(8,R6),8(R8)
*
*********************************************************************
*  SET UP MESSAGE BLOCK FOR TRACE ENTRY FOR CHANGED NAME            *
*********************************************************************
*
         MVC   MESS1,MESS1T            SET UP MESSAGE
         MVC   NEWPSB,8(R8)            NEW PSB NAME
         MVC   OLDPSB,0(R8)            OLD PSB NAME
         B     TRACE                   GO PUT TRACE ENTRY
*
*********************************************************************
* SET UP MESSAGE BLOCK FOR TRACE ENTRY FOR PSB NOT FOUND            *
* SETUP THE NORMAL RETURN CODE                                      *
*********************************************************************
*
NOTFOUND EQU   *
         MVC   0(8,R6),DUMMYPSB
         MVC   MESS1,MESS2T            SET UP MESSAGE
         MVC   OLDPSB,0(R6)            SUPPLIED PSB NAME
         MVC   NEWPSB,=CL8''           CLEAR FIELD
         LA    R1,UERCNORM             SET UP NORMAL RETURN CODE
         B     TRACE                   GO PUT TRACE ENTRY
*
*********************************************************************
*  Issue trace put macro                                            *
*********************************************************************
*
TRACE    EQU   *
         LA    R6,MESSAGE              STORE ADDRESS...
         ST    R6,MESSAGEA             ...INTO BLOCK DESCRIPTOR
         LA    R6,L'MESSAGE            STORE LENGTH...
         ST    R6,MESSAGEL             ...INTO BLOCK DESCRIPTOR
         LA    R8,384                  SET UP TRACE-ID
*
```

Figure 37. Example of XDLIPRE user exit to change PSB names 4/6

```
         DROP  R5                      REUSE R5 TO MAP DFHTRPT
         USING DFHTRPT_ARG,R5          XPI PARAMETER LIST
*
         DFHTRPTX CALL,                                                X
               CLEAR,                                                  X
               IN,                                                     X
               FUNCTION(TRACE_PUT),                                    X
               POINT_ID((R8)),                                         X
               DATA1(MESSAGEA,MESSAGEL),                               X
               OUT,                                                    X
               RESPONSE(*)
*
*********************************************************************
*When the rest of the exit program is complete, free the storage   *
*and return.                                                        *
*********************************************************************
*
      DROP  R5                   REUSE REGISTER 5 TO MAP DFHSMMC
      USING DFHSMMC_ARG,R5       XPI PARAMETER LIST
*
*********************************************************************
*  Issue the DFHSMMCX macro call                                    *
*  Store the return code in register 6                              *
*********************************************************************
*
          L     R6,RETCODE              PICK UP SAVED RETURN CODE
*
          DFHSMMCX CALL,                                             X
                CLEAR,                                               X
                IN,                                                  X
                FUNCTION(FREEMAIN),                                  X
                ADDRESS((R7)),                                       X
                STORAGE_CLASS(USER),                                 X
                OUT,                                                 X
                RESPONSE(*),                                         X
                REASON(*)
*
*********************************************************************
*Restore registers, set return code, and return to user exit handler*
*********************************************************************
*
          L     R13,UEPEPSA
          ST    R6,16(13)               STORE INTO R15 SLOT OF SA
          RETURN (14,12)
*
*********************************************************************
*old and new PSB names, in pairs                                    *
*********************************************************************
*
```

Figure 38. Example of XDLIPRE user exit to change PSB names 5/6

```
PSBS     EQU   *
          DC    CL8'PC3CONEW'            VALID
          DC    CL8'PC3CONE2'            VALID
          DC    CL8'PC3FRED'             INVALID
          DC    CL8'PC3CONEW'            VALID
          DC    CL8'PC3JOE'              INVALID
          DC    CL8'PC3JOEX'             INVALID
*
MESS1T    DC    CL21' HAS BEEN CHANGED TO '
MESS2T    DC    CL21' WAS NOT FOUND'
MESS3T    DC    CL37'THIS WAS NOT A DLI SCHEDULE CALL'
DUMMYPSB  DC    CL8' '
          LTORG
          END   DLIPR
```

Figure 39. Example of XDLIPRE user exit to change PSB names 6/6

Summary of DBCTL abends and return codes

With DBCTL, your program specification block (PSB) scheduling request might fail either because DBCTL is not available, or because the PSB could not be found.

However, after a successful PSB schedule, CICS might be disconnected from DBCTL for some reason, and subsequent DBCTL requests will fail. This situation, which is unique to a DBCTL environment, causes an ADCJ abend to be issued. Table 12 summarizes the schedule failure codes and abends in a DBCTL environment, and the conditions that can arise on a PSB schedule request because DBCTL is not available or the PSB cannot be found.

Table 12. Summary of abends and return codes

Request	EXEC abend	CALL UIBDLTR	CALL UIBFCTR	CALL abend	Explanation
PSB schedule or DL/I request	ADCA	----	----	ADCA	Error detected in DBCTL.
DL/I request	ADCB	----	----	ADCB	PSB not scheduled.
PSB schedule request	ADCC	----	----	ADCC	PSB already scheduled detected in DBCTL.
DL/I request	ADCD	----	----	ADCD	Deadlock detected.
PSB schedule or DL/I request	ADCE	----	----	ADCE	Bad response code has been returned from DFHDBAT.
DL/I request	ADCI	----	----	ADCI	Lock outstanding.
DL/I request	ADCJ	----	----	ADCJ	DBCTL not available on DL/I request[1].
PSB schedule or DL/I request	ADCN	----	----	ADCN	FORCEPURGE issued while running in DBCTL.
PSB schedule request	ADCP	----	----	ADCP	The user is not authorized to use the PSB.
PSB schedule request	ADCQ	----	----	ADCQ	The SYSSERVE keyword or the I/O PCB option was not specified, and the PSB does not contain any DB PCBs.
DL/I request	ADCR	----	----	ADCR	DL/I request (other than PSB schedule) issued when DBCTL not connected.

Table 12. Summary of abends and return codes (continued)

Request	EXEC abend	CALL UIBDLTR	CALL UIBFCTR	CALL abend	Explanation
PSB schedule request	ADDA	----	----	ADDA	An error response from the storage domain.
PSB schedule or DL/I request	ADDK	----	----	ADDK	CICS Lock manager call failed.
Terminate request	ASPR	----	----	ASPR	Single-phase commit request issued but CICS unable to report outcome. IMS updates are either backed out, or committed. IMS is not indoubt about the UOW.
Terminate request	ASP7	----	----	ASP7	Single-phase commit request failed. IMS backed out any updates in the UOW.
PSB schedule request	DHTA	X'01' (PSBNF)	X'08' (INVREQ)	----	PSB not found[2].
PSB schedule request	DHTC	X'03' (PSBSCH)	X'08' (INVREQ)	----	PSB already scheduled detected in CICS.
PSB schedule request	DHTE	X'05' (PSBFAIL)	X'08' (INVREQ)	----	PSB initialization failed.
Terminate request	DHTG	X'07' (TERMNS)	X'08' (INVREQ)	----	PSB not scheduled.
DL/I request	DHTH	X'08' (FUNCNS)	X'08' (INVREQ)	----	PSB not scheduled, detected by CICS.
PSB schedule request	DHTJ	X'FF' (DLINA)	X'08' (INVREQ)	----	DBCTL not available on PSB scheduling[3].
PSB schedule, DL/I, and terminate requests	DHxx	----	----	----	Many reasons. xx is the PCB status code. (See also "Preventing DHxx abends after EXEC DLI SCHD PSB failure" on page 101.)
PSB schedule or DL/I request	----	X'00' (INVARG)	X'08' (INVREQ)	----	Invalid argument.
PSB schedule or DL/I request	---- TR status code in DIB-STAT	X'04' (NOTDONE)	X'08' (INVREQ)	----	Global user exit XDLIPRE indicates that DL/I request should not be run.

Note:

1. DBCTL is in use, and a PSB has been scheduled. However, the connection between CICS and DBCTL has since been broken.
2. The PSB was not found in PDIR and DBCTL was not ready. Alternatively, the PSB was not found in PDIR and DBCTL was ready but the PSB was not found in DBCTL APPLCTN.
3. DBCTL was not ready at the time of the DL/I request.

See *IMS Application Programming: EXEC DLI Commands* for details of DL/I status codes.

If you use remote DL/I with DBCTL, you might also receive Axxx and DHxx abends not listed here. For information about DHxx abends (where 'xx' indicates the DL/I status code), see *IMS Application Programming: EXEC DLI Commands*.

Chapter 6. Security checking with DBCTL

Covers the different types of security checking you might need.

When using CICS with DBCTL, you might want to use one or more of the following ***optional*** security facilities:

- "PSB authorization checking by CICS"
- Resource access security checking by DBCTL
- DBCTL password security checking

For details on resource access security checking by DBCTL and DBCTL password security checking see *IMS System Administration Guide* or *IMS Administration Guide: System*.

Of the resources you can protect using IMS security, you need be concerned only with PSBs, databases, and commands.

PSB authorization checking by CICS

At PSB scheduling time, CICS invokes security checking to find out whether the terminal user is authorized to access the PSB. The actual check is carried out by an external security manager, which can be RACF or your own security program.

Although PSB scheduling requests are sent to DBCTL for processing, CICS does PSB authorization checking. See the *CICS Customization Guide* for programming information about writing your own security program.

Chapter 7. Problem determination for DBCTL

In a CICS-DBCTL environment, you need to correlate information produced by the CICS system with information produced by the DBCTL system. This information includes trace entries produced by CICS and DBCTL, dumps produced by CICS, the DRA, and DBCTL, and messages produced by CICS, the DRA, and DBCTL.

The link between CICS and DBCTL in all the above cases is the recovery token. It appears in trace entries, in dumps (including the dump header), and in messages issued by CICS and DBCTL.

See the *CICS Problem Determination Guide* for more detailed help on dealing with problems, beginning from symptoms through to identification and solution. For detailed component descriptions of DBCTL, which you may find useful in debugging, see the *CICS Diagnosis Reference* manual. See *IMS Messages and Codes* for similar guidance on messages and abend codes issued by the DRA and by DBCTL.

Interactions between CICS and DBCTL

Errors can occur during interactions between CICS and DBCTL at the interface level or during interactions between CICS and DBCTL caused by requests.

Interactions between CICS and DBCTL at the interface level

- Connection to DBCTL.

  See "Connection to DBCTL has failed to complete" on page 120.
- Disconnection from DBCTL. (This includes intentional operator-requested disconnection, and unintentional disconnections caused by failures of the system, or parts of the CICS-DBCTL interface.)

  See "Disconnection from DBCTL has failed to complete" on page 120.

Interactions between CICS and DBCTL caused by requests

- Requests that are issued by applications:
  - Waits or failures during PSB scheduling.

    See "Failures during PSB scheduling" on page 121.
  - Waits or failures during the processing of a DL/I request.

    See "Failures during DL/I request processing" on page 121.
- Requests that are issued as a result of task termination, including syncpoint processing:
  - Failures during PREPARE processing
  - Failures during COMMIT processing (TERM call or task termination)
  - Failures during resynchronization of UOWs

  In all these cases, see "Thread termination" on page 131.

DBCTL error scenarios

DBCTL errors can occur in a variety of ways, such as during connection to DBCTL, or during PSB scheduling. Use dumps and trace messages to help diagnose the error for fixing.

Connection to DBCTL has failed to complete

In this situation, the DRA may be in a "wait" state because you attempted to connect CICS to DBCTL using the CDBC transaction, but the connection process failed to complete.

Connection to DBCTL using the CICS-supplied transaction CDBC takes place in two phases. In phase 1, CDBC passes the request for connection to IMS and returns. In phase 2, IMS processes the request asynchronously and returns to CICS when connection is complete. To discover where the problem occurred, try to find out how far the connection attempt has progressed by:

- Pressing PF2 on the CDBC menu panel to refresh this display, as described in "CDBC transaction for connect and disconnect" on page 37; or
- Using the CDBI inquiry panel, as described in "CDBI transaction for inquiry" on page 41.

If connection is in phase 1, the following message is issued:

```
DFHDB8291 I DBCTL CONNECT PHASE 1 IN PROGRESS
```

It is very unlikely that a wait will occur during this phase, unless there is a problem with the CICS transaction.

If connection is in phase 2, the following message is issued:

```
DFHDB8292 I DBCTL CONNECT PHASE 2 IN PROGRESS
```

If phase 2 fails to complete, the failure is associated with IMS. This may be because:

- The DRA startup table is pointing to the wrong system because the DBCTL subsystem ID is incorrect. If this is so, CICS issues a WTO message saying:

  ```
  SUBSYSTEM xxxx NOT ACTIVE.  REPLY WAIT OR CANCEL
  ```

  where xxxx is the subsystem ID indicated on the CDBC panel.

  See "Defining the IMS DRA startup parameter table" on page 27 for information on specifying the DBCTL subsystem ID.
- DBCTL has been initialized, but no restart command has been issued. Remember that DBCTL needs a restart command unless you are using AUTO start. See "Connecting to DBCTL: overview" on page 35 and "Restarting DBCTL" on page 66 for information on restarting DBCTL and on the implications of different restart options.

If neither of the above situations applies, the problem is in IMS; see *IMS Diagnosis Guide and Reference* for further guidance.

For an example of the trace entries produced by CICS for a successful connection to DBCTL, see "Connection to DBCTL" on page 123.

Disconnection from DBCTL has failed to complete

In this case, the DRA may be in a wait state because you attempted to disconnect CICS from DBCTL using the CDBC transaction, but the disconnection process failed to complete.

For an example of the trace entries produced by CICS for a successful disconnection from DBCTL, see "Disconnection from DBCTL" on page 126.

When you use CDBC to disconnect from DBCTL, it invokes another CICS transaction, CDBT. CDBT makes the disconnection request to DBCTL, and is suspended by CICS while DBCTL services the request asynchronously.

If disconnection fails to complete, you can inquire on CDBT using CEMT INQ TASK to see how far disconnection has progressed. You will probably find CDBT is waiting on resource name DLSUSPND and resource type DBCTL, which means the request is being processed by DBCTL. For an illustrated example, see the description of CEMT INQ TASK in "Purging a transaction that is using DBCTL" on page 60.

- If CDBT is waiting on DLSUSPND, what you do next depends on whether the disconnection requested was orderly or immediate. (Use the CDBI inquiry panel, as described in "CDBI transaction for inquiry" on page 41, if you need to find out.)
  - If you have requested orderly disconnection, it is likely that DBCTL is waiting for a task issuing many DL/I requests, or for a conversational task, perhaps one that is waiting for input from an unattended terminal.

    You can, if necessary, override an orderly disconnection by requesting immediate disconnection, in which case the process should conclude at once. However, be aware that immediate disconnection can cause indoubt UOWs, and leave database records unavailable to other CICS systems using that DBCTL until it is reconnected, as described in "Deciding whether to use orderly or immediate disconnection" on page 41.
  - If you have requested immediate disconnection, and it has not taken place, it is likely that an unexpected wait within IMS has occurred. See *IMS Diagnosis Guide and Reference* for further guidance.
- If CDBT is not waiting on DLSUSPND, this indicates a problem in CICS. See the the *CICS Problem Determination Guide* for information on dealing with it.

Failures during PSB scheduling

For examples of trace entries produced by CICS during PSB scheduling (both successful and failed), see "PSB schedule" on page 128 and "PSB scheduling failure" on page 129.

Use the DBCTL operator command /DISPLAY as follows:

- /DISPLAY PROGRAM psbname to check that the ACB is valid. A status of "invalid" means that the PSB was not defined during IMS system generation. A status of "notinit" means that the ACB is not in the ACBLIB. A status of "stopped" means an error has caused DBCTL to stop the PSB, or that a /STOP command has been issued for the PSB. Investigate the cause of this error. When resolved, use /START PROGRAM psbname to start the PSB again.
- /DISPLAY DATABASE dbname to check that the databases are valid.

Failures during DL/I request processing

The DRA may have entered a "wait" state because you have a CICS task in a wait state.

For an example of the trace entries produced by CICS during DL/I request processing, see "CICS task issuing DL/I requests to be processed by DBCTL" on page 130. For an example of the trace entries produced by DBCTL during DL/I request processing, see "Trace entries produced by DBCTL" on page 132.

If a task appears to have "hung", query it using CEMT INQ TASK, as for any CICS task. If you have a task waiting on a resource name of DLSUSPND and resource type DBCTL, the task has made a DL/I request and has been suspended in CICS while DBCTL services that request. If repeated use of CEMT INQ TASK shows the task still waiting on DLSUSPND, it has "hung" in DBCTL. If you want to purge the task, you must use DBCTL operator commands to do so. See "Purging a transaction that is using DBCTL" on page 60 for an illustrated example of using CEMT INQ TASK and the relevant DBCTL operator commands in this way.

If the task is not waiting on DLSUSPND, this may indicate a problem in CICS. See the *CICS Problem Determination Guide* for information about dealing with it.

Correlating activity in DBCTL and CICS

Using the /DISPLAY command to display DBCTL activity and the CEMT INQ TASK to display CICS activity are useful means of correlating what is happening on each side of the interface.

Check to see that the recovery token matches in CICS and DBCTL. If it does not, this may indicate a thread hanging. /DISPLAY CCTL ALL displays all the threads associated with CICS tasks in DBCTL. If you enter /DISPLAY ACTIVE ALL, region and DC activity is also displayed, enabling you to find out if a BMP is waiting in DBCTL.

Trace for CICS DBCTL

When examining traces entries produced by CICS and DBCTL, you must relate them according to whether they are produced at the same time in CICS and in DBCTL, or at different times. You must also know how to find the relevant parts of each trace and use them to correlate what is happening in CICS and in DBCTL.

Trace entries produced by CICS

Use the CICS-supplied transaction CETR to trace DBCTL activity. CETR traces DL/I requests until they leave DFHDBAT.

The sections that follow give examples of CICS trace entries produced at the following points:

- "Connection to DBCTL" on page 123
- "Disconnection from DBCTL" on page 126
- "PSB schedule" on page 128
- "PSB scheduling failure" on page 129
- "CICS task issuing DL/I requests to be processed by DBCTL" on page 130
- "Thread termination" on page 131

See *CICS Problem Determination Guide* for details of the general format of CICS trace entries, how to select trace options for component and task tracing, whether to use "standard" or "special" tracing, and how to start and stop tracing selectively. See the *CICS Operations and Utilities Guide* for help on formatting and printing trace entries, including a sample job you can use to do so.

The numbers in the margin to the left of the example traces point to things that you may find useful in correlating CICS and DBCTL activity, but please note that these additional numbers are ***not*** part of the trace output. Also note that we have omitted some trace entries for brevity. This is indicated by the following symbol:

.
.

Connection to DBCTL

Figure 40 on page 124 shows an example of the CICS trace entries produced when CICS connects to DBCTL.

```
1                                                .
                                                 .
2     00028 1 AP 00E1 EIP   ENTRY LINK                                          0004,07301464 ....,08000E02 ....
      00028 1 PG 1101 PGLE  ENTRY LINK_EXEC             DFHDBCON,07301088 , 00000014
      00028 1 DD 0301 DDLO  ENTRY LOCATE                06D08F80,07301698,PPT,DFHDBCON
      00028 1 DD 0302 DDLO  EXIT  LOCATE/OK             D7D7E3C5 , 06D89858
      00028 1 LD 0001 LDLD  ENTRY ACQUIRE_PROGRAM       06D8BF50
                                                 .
                                                 .
3     00028 1 XM 1101 XMAT  ENTRY ATTACH                CDBO,07302E38 , 00000004,0,NONE,C,NO,YES,NO,0
      00028 1 XM 0401 XMLD  ENTRY LOCATE_AND_LOCK_TRANDEF CDBO
      00028 1 DD 0301 DDLO  ENTRY LOCATE                06D00040,07303314,TXD,CDBO
      00028 1 DD 0302 DDLO  EXIT  LOCATE/OK             06D86B78 , D7000000
                                                 .
                                                 .
4     00028 1 LD 0001 LDLD  ENTRY ACQUIRE_PROGRAM       DFHDBSPX,YES
      00028 1 LD 0002 LDLD  EXIT  ACQUIRE_PROGRAM/OK    870A0020,070A0000
                                                 .
                                                 .
5     00028 1 AP 00E1 EIP   ENTRY ENABLE                                        0004,07302AD4 ...M,08002202 ....
                                                 .
6    1CICS/ESA - AUXILIARY TRACE FROM 07/20/95 - APPLID CICSKPG1 - TIME OF FIRST ENTRY ON THIS PAGE 11:26:58.7144860002
                                                 .
7     00028 1 AP 2522 ERM   EVENT PASSING-CONTROL-TO-TRUE(DBCTL   )
      00028 1 AP 0310 DBAT  ENTRY APPLICATION           REQUEST
8     00028 1 AP 0314 DBAT  EVENT DRA-ROUTER-LOAD       , LOAD-RESPONSE-CODE (00000000)
9     00028 1 AP 0315 DBAT  EVENT ABOUT-TO-INVOKE-DRA   FOR INTERFACE REQUEST , 0100
10    00028 1 AP 0316 DBAT  EVENT RECEIVES-CONTROL-FROM-DRA FOR INTERFACE REQUEST , 00000000
      00028 1 AP 0313 DBAT  EXIT  DBAT-RESPONSE-CODE    (00000000)
11    00028 1 AP 2523 ERM   EVENT REGAINING-CONTROL-FROM-TRUE(DBCTL   )
      00028 1 AP 2521 ERM   EXIT  APPLICATION-CALL-TO-TRUE(DBCTL   )
12    00028 1 ME 0301 MEME  ENTRY SEND_MESSAGE          1FB4,073D642C , 00000004,073D5060 , 00000002,DB
      00028 1 ME 0501 MEIN  ENTRY INQUIRE_MESSAGE_DATA  86BB5AE0,DFHMET1E,1FB4,073039CD , 00000000 , 0000001C,07303967 , 00000000
      00028 1 KE 0101 KETI  ENTRY INQ_LOCAL_DATETIME_DECIMAL
      00028 1 KE 0102 KETI  EXIT  INQ_LOCAL_DATETIME_DECIMAL/OK 07201995,095757,097993,MMDDYYYY
      00028 1 KE 0401 KEGD  ENTRY INQUIRE_KERNEL
      00028 1 KE 0402 KEGD  EXIT  INQUIRE_KERNEL/OK     CICSKPG1,CIA1
      00028 1 ME 0502 MEIN  EXIT  INQUIRE_MESSAGE_DATA/OK 06BB5D7C,06BC5E07,06BC5E1D,06BC5E7C,,I,095757,20071995,M,CIA1,CICSKPG1
      00028 1 ME 0312 MEME  EVENT ISSUE-MVS-GETMAIN
      00028 1 ME 0313 MEME  EVENT MVS-GETMAIN-COMPLETE
13    00028 1 DU 0500 DUDT  ENTRY INQUIRE_SYSTEM_DUMPCODE DB8116
      00028 1 DU 0600 DUTM  ENTRY INQUIRE_SYSTEM_DUMPCODE DB8116
      00028 1 DU 0601 DUTM  EXIT  INQUIRE_SYSTEM_DUMPCODE/EXCEPTION DUMPCODE_NOT_FOUND,0,0,,,,
      00028 1 DU 0501 DUDT  EXIT  INQUIRE_SYSTEM_DUMPCODE/EXCEPTION DUMPCODE_NOT_FOUND,0,0,,,,
      00028 1 ME 0401 MEBU  ENTRY BUILD_MESSAGE         06BC5E07,06BB5D7C,20071995,M,095757,CIA1,CICSKPG1,0730369D , 00000009,073
      00028 1 ME 0402 MEBU  EXIT  BUILD_MESSAGE/OK      0
      00028 1 ME FF35 MEFO  ENTRY -FUNCTION(FORMAT_MESSAGE) 0698B390 , 0000006F,1,78,073039EB , 00000001,YES
      00028 1 ME FF36 MEFO  EXIT  -FUNCTION(FORMAT_MESSAGE) OK
14    00028 1 AP F600 TDA   ENTRY WRITE_TRANSIENT_DATA  CDBC,073039FB , 00000001,NO
                                                 .
                                                 .
15    00028 1 DU 0500 DUDT  ENTRY INQUIRE_SYSTEM_DUMPCODE DB8210
                                                 .
                                                 .
16    00028 1 DU 0500 DUDT  ENTRY INQUIRE_SYSTEM_DUMPCODE DB8292
                                                 .
17                                               .
      00038 1 DS 0005 DSSR  EXIT  WAIT_MVS/OK
18    00038 1 AP 0306 DBCT  EVENT POSTED                FOR CONNECTION COMPLETE
19    00038 1 ME 0301 MEME  ENTRY SEND_MESSAGE          1FA5,0698B240 , 00000004,073D5060 , 00000002,DB
                                                 .
                                                 .
      00038 1 DU 0500 DUDT  ENTRY INQUIRE_SYSTEM_DUMPCODE DB8101
                                                 .
                                                 .
20    00038 1 GC 2010 CCCC  ENTRY WRITE                 00108194 , 00000008,DBCTL,STATUS
                                                 .
                                                 .
      00038 1 GC 2050 CCCC  EXIT  WRITE/OK
21    00038 1 PG 0A01 PGLU  ENTRY LINK_URM              DFHDBUEX,001081F0 , 0000000B,NO
      00038 1 DD 0301 DDLO  ENTRY LOCATE                06D08F80,00108220,PPT,DFHDBUEX
      00038 1 DD 0302 DDLO  EXIT  LOCATE/OK             D7D7E3C5 , 06D89A50
                                                 .
                                                 .
22    00038 1 AP 0064 USER  EVENT APPLICATION-PROGRAM-ENTRY CONNECT DBCTL HAS JUST BEEN CONNECTED
                                                 .
                                                 .
      00038 1 AP 1941 APLI  EXIT  START_PROGRAM/OK       ....,DFHDBUEX
      00038 1 LD 0001 LDLD  ENTRY RELEASE_PROGRAM       0732B450,86D5B028
      00038 1 LD 0002 LDLD  EXIT  RELEASE_PROGRAM/OK    06D5B000,3A8,ECDSA
      00038 1 PG 0A02 PGLU  EXIT  LINK_URM/OK
      00038 1 AP 00E1 EIP   ENTRY RESYNC                                        0004,001087C4 ..gD,08001604 ....
```

Figure 40. CICS trace entries produced during connection to DBCTL 1 of 2

```
22    00038 1 AP 0064 USER  EVENT APPLICATION-PROGRAM-ENTRY CONNECT DBCTL HAS JUST BEEN CONNECTED
                                                .
                                                .
      00038 1 AP 1941 APLI  EXIT  START_PROGRAM/OK      ....,DFHDBUEX
      00038 1 LD 0001 LDLD  ENTRY RELEASE_PROGRAM       0732B450,86D5B028
      00038 1 LD 0002 LDLD  EXIT  RELEASE_PROGRAM/OK    06D5B000,3A8,ECDSA
      00038 1 PG 0A02 PGLU  EXIT  LINK_URM/OK
      00038 1 AP 00E1 EIP   ENTRY RESYNC                                          0004,001087C4 ..gD,08001604 ....
                                                .
                                                .
23    00038 1 AP E161 EXEC  EXIT  RESYNC                'DBCTL   ' AT X'0713F062','JB1A    ' AT X'8698B270',AT X'00000000',0 AT X
      00038 1 AP E111 EISR  EXIT  TRACE_EXIT/OK
      00038 1 AP 00E1 EIP   EXIT  RESYNC                OK                        00F4,00000000 ....,00001604 ....
      00038 1 AP 00E1 EIP   ENTRY SYNCPOINT                                       0004,001087C4 ..gD,08001602 ....
                                                .
                                                .
      00038 1 AP E161 EXEC  EXIT  SYNCPOINT             0,0,ASM,09490000
                                                .
                                                .
24    00028 1 ME 0301 MEME  ENTRY RETRIEVE_MESSAGE      2065,000550A7 , 00000000 , 00000033,E,DB
      00028 1 ME 0501 MEIN  ENTRY INQUIRE_MESSAGE_DATA  86BB5AE0,DFHMET1E,2065,07301F95 , 00000000 , 0000001C,07301F2F , 00000000
                                                .
                                                .
      00028 1 ME 0502 MEIN  EXIT  INQUIRE_MESSAGE_DATA/OK 06BB5D7C,06BC7416,06BC742C,06BC744D,I,,095759,20071995,M,CIA1,CICSKPG1
      00028 1 DU 0500 DUDT  ENTRY INQUIRE_SYSTEM_DUMPCODE DB8293
```

Figure 41. CICS trace entries produced during connection to DBCTL 2 of 2

Note:

1. Phase 1 of connection begins.
2. Locating DFHDBCON and loading if not already loaded. (In this example, CICS and DBCTL have already been connected during this CICS session, so DFHDBCON has already been loaded.)
3. The control transaction, CDBO, is attached. CDBO enables the DRA to pass information from itself and DBCTL independently of CICS. It is invoked whenever the DRA needs to determine whether to continue processing, which is when:
   - The DRA has successfully connected to DBCTL
   - DBCTL has been terminated normally using /CHECKPOINT FREEZE or /CHECKPOINT PURGE
   - Connection to DBCTL has failed
   - A CICS request to connect to DBCTL has been canceled
   - The DRA fails
   - DBCTL fails
4. Loading programs needed: DFHDBSPX (shown in example), plus DFHDBCX, DFHDBMOX, DFHDBREX, DFHDBSTX, DFHDBSSX, DFHDBTOX, and DFHDBAT.
5. DFHDBCON enables DFHDBAT.
6. A timestamp is included in the header line of every page of CICS abbreviated auxiliary trace output to help you match trace entries with external events.
7. DFHERM invokes DFHDBAT for connection request.
8. DRA router module DFSPRRC0 loaded.
9. DRA is invoked for interface request. The type of interface request is indicated by request type from the PAPL: 0100 is a CONNECT request. (See “PAPL request and return codes” on page 138.)
10. DBCTL return code (00000000). See “Return codes in DBCTL” on page 137.
11. Control is passed back to DFHERM.

12. Phase 1 of connection has ended at this point. Message DFHDB8116 is issued confirming that connection is proceeding. The message includes the DBCTL identifier and the DRA suffix used.
13. When a message has been issued, the CICS dump domain checks to see if the user has requested any action for that message (using the CEMT SET SYDUMPCODE, or the EXEC CICS SET SYSDUMPCODE commands, (In this case, no dump has been requested, as indicated by DUMPCODE_NOT_FOUND.) However, when you are using abbreviated trace, entries such as INQUIRE_SYSTEM_DUMPCODE DB8116 (in which the system dump code is the message number without the characters "DFH") are useful in indicating which messages have been issued. (Complete message numbers are included in full trace.)
14. Message DFHDB8116 is sent to transient data destination CDBC.
15. Message DFHDB8210 is issued confirming that connection to DBCTL is proceeding.
16. Message DFHDB8292 is issued indicating that CICS is in phase 2 of connecting to DBCTL.
17. At this point, DBCTL exits are loaded, which causes I/O activity. The task is suspended, and the control transaction, CDBO, starts. This is indicated by the task number changing (from 00031 to 00032). Control transaction enters a series of waits. CDBO invokes the CICS-DBCTL interface control program (DFHDBCT).
18. DBCTL notifies CICS that CICS-DBCTL connection is complete.
19. Message DFHDB8101 is issued.
20. A record is written to the global catalog, indicating which DBCTL should be reconnected to if there is a CICS failure. (See "Program list table (PLT)" on page 16 and "Connecting DBCTL to CICS automatically" on page 36.)
21. DFHDBUEX, the CICS-supplied user replaceable program for use with DBCTL, is linked. Trace entries following invocation of DFHDBUEX depend on what you have coded in your own version. (See "DFHDBUEX" on page 31.)
22. In this example, the user has coded DFHDBUEX to issue a trace entry stating that DBCTL has just been connected.
23. CICS issues an EXEC CICS RESYNC command to resynchronize any outstanding DBCTL indoubt UOWs. (See Chapter 4, "Recovery and restart operations for DBCTL," on page 65.)
24. Control transaction waits have ended: task number changes back again (from 00032 to 00031). Message DFHDB8293 is issued confirming that DBCTL is connected and ready.

Disconnection from DBCTL

This table shows some examples of CICS trace entries produced at disconnection from DBCTL.

```
1    1CICS/ESA - AUXILIARY TRACE FROM 07/20/95 - APPLID CICSKPG1 - TIME OF FIRST ENTRY ON THIS PAGE 11:26:58.7144860002
                                                .
2                                               .
      00047 1 AP 00E1 EIP   ENTRY START                                          0004,07301464 ....,08001008 ....
3     00047 1 XM 0401 XMLD  ENTRY LOCATE_AND_LOCK_TRANDEF CDBT
      00047 1 DD 0301 DDLO  ENTRY LOCATE               06D00040,07301820,TXD,CDBT
      00047 1 DD 0302 DDLO  EXIT  LOCATE/OK            06D86C10 , D7000000
                                                .
                                                .
4     00047 1 DU 0500 DUDT  ENTRY INQUIRE_SYSTEM_DUMPCODE DB8211
                                                .
                                                .
5     00047 1 DU 0500 DUDT  ENTRY INQUIRE_SYSTEM_DUMPCODE DB8294
                                                .
                                                .
6     00048 1 PG 0901 PGPG  ENTRY INITIAL_LINK         DFHDBDSC
                                                .
                                                .
7     00048 1 AP 00E1 EIP   ENTRY ADDRESS                                        0004,0005B010 ....,08000202 ....
                                                .
                                                .
8     00048 1 PG 0A01 PGLU  ENTRY LINK_URM             DFHDBUEX,0005B0C4 , 0000000B,NO
      00048 1 DD 0301 DDLO  ENTRY LOCATE               06D08F80,0005B3A4,PPT,DFHDBUEX
      00048 1 DD 0302 DDLO  EXIT  LOCATE/OK            D7D7E3C5 , 06D89A50
      00048 1 LD 0001 LDLD  ENTRY ACQUIRE_PROGRAM      0732B450
      00048 1 LD 0002 LDLD  EXIT  ACQUIRE_PROGRAM/OK   86D5B028,06D5B000,3A8,0,REUSABLE,ECDSA,OLD_COPY
9     00048 1 AP 1940 APLI  ENTRY START_PROGRAM        DFHDBUEX,NOCEDF,FULLAPI,URM,NO,07309828,0005B0C4 , 0000000B,2
                                                .
                                                .
      00048 1 AP 0065 USER  EVENT APPLICATION-PROGRAM-ENTRY DISCONN DBCTL HAS JUST BEEN DISCONNECTED
                                                .
                                                .
10    00048 1 LD 0001 LDLD  ENTRY RELEASE_PROGRAM      0732B450,86D5B028
      00048 1 LD 0002 LDLD  EXIT  RELEASE_PROGRAM/OK   06D5B000,3A8,ECDSA
      00048 1 PG 0A02 PGLU  EXIT  LINK_URM/OK
      00048 1 AP 2520 ERM   ENTRY APPLICATION-CALL-TO-TRUE(DBCTL   )
                                                .
                                                .
      00048 1 AP 2522 ERM   EVENT PASSING-CONTROL-TO-TRUE(DBCTL   )
      00048 1 AP 0310 DBAT  ENTRY APPLICATION          REQUEST
11    00048 1 AP 0315 DBAT  EVENT ABOUT-TO-INVOKE-DRA  FOR INTERFACE REQUEST , 0400
12    00048 1 AP 0304 DBSPX EVENT ABOUT-TO-ISSUE-WAIT  FOR DISCONNECTION REQUEST
      00048 1 DS 0004 DSSR  ENTRY WAIT_MVS             DLSUSPND,DBCTL,0005B444,NO,OTHER_PRODUCT
      00048 1 DS 0005 DSSR  EXIT  WAIT_MVS/OK
      00048 1 AP 0305 DBSPX EVENT POSTED               FOR DISCONNECTION REQUEST
13    00048 1 AP 0316 DBAT  EVENT RECEIVES-CONTROL-FROM-DRA FOR INTERFACE REQUEST , 00000000
                                                .
                                                .
14    00048 1 ST 0003 STST  ENTRY RECORD_STATISTICS    072F7618 , 00000054,USS
                                                .
                                                .
      00048 1 ST 0004 STST  EXIT  RECORD_STATISTICS/OK
      00048 1 AP 0313 DBAT  EXIT  DBAT-RESPONSE-CODE   (00000000)
      00048 1 AP 2523 ERM   EVENT REGAINING-CONTROL-FROM-TRUE(DBCTL   )
      00048 1 AP 2521 ERM   EXIT  APPLICATION-CALL-TO-TRUE(DBCTL   )
15    00048 1 GC 2010 CCCC  ENTRY WRITE                0005B0BC , 00000008,DBCTL,STATUS
16                                              .
                                                .
      00048 1 DS 0004 DSSR  ENTRY WAIT_MVS             ASYNRESP,CCVSAMWT,06C8D5C0,NO,IO
      00038 1 DS 0005 DSSR  EXIT  WAIT_MVS/OK
17    00038 1 AP 0306 DBCT  *EXC* EVENT                POSTED FOR DFHDBCT SHOULD TERMINATE
      00038 1 AP 00E1 EIP   ENTRY START                                          0004,001087C4 ..gD,08001008 ....
                                                .
                                                .
      00038 1 XM 0401 XMLD  ENTRY LOCATE_AND_LOCK_TRANDEF CDBD
      00038 1 DD 0301 DDLO  ENTRY LOCATE               06D00040,0730C078,TXD,CDBD
      00038 1 DD 0302 DDLO  EXIT  LOCATE/OK            06D86918 , D7000000
                                                .
                                                .
      00038 1 AP 00F3 ICP   ENTRY INITIATE             CDBD                      4003,0000000C ....,00000000 ....,CDBD
                                                .
                                                .
18    00049 1 LD 0001 LDLD  ENTRY RELEASE_PROGRAM      DFHDBSSX,8711A910
      00049 1 LD 0002 LDLD  EXIT  RELEASE_PROGRAM/OK
                                                .
                                                .
      00049 1 ME 0502 MEIN  EXIT  INQUIRE_MESSAGE_DATA/OK 06BB5D7C,06BC56B8,06BC56CE,06BC5710,,I,100011,20071995,M,CIA1,CICSKPG1
19    00049 1 DU 0500 DUDT  ENTRY INQUIRE_SYSTEM_DUMPCODE DB8102
```

Figure 42. CICS trace entries produced during disconnection from DBCTL

Note:

1. Timestamp, as mentioned in “Connection to DBCTL” on page 123.
2. Phase 1 of disconnection begins at this stage.
3. The CICS-DBCTL interface disconnection transaction, CDBT, is attached.
4. Message DFHDB8211 is issued to confirm that orderly disconnection is proceeding. This message is issued in response to the user pressing PF5 on the CDBC screen. (For an immediate disconnection, message DFHDB8212 is issued.)
5. Message DFHDB8294 is issued confirming that orderly disconnection is in progress. (If immediate disconnection had been requested, message DFHDB8295 would have been issued.)
6. CDBT invokes CICS-DBCTL interface disconnection program, DFHDBDSC. A wait is entered (task number changes, from 00034 to 00035).
7. The EXEC interface program, DFHEIP, links to the CICS-DBCTL user-replaceable program, DFHDBUEX.
8. DFHDBUEX is loaded.
9. Trace entries at this point depend on what, if anything, you have coded in your own version of DFHDBUEX. (See “DFHDBUEX” on page 31.) In this example, DFHDBUEX has been coded to issue a trace entry stating that DBCTL has just been disconnected.
10. DFHDBUEX is released and control is passed back to DFHDBDSC.
11. The DRA is invoked for an interface request. (PAPL request type 0400 indicates the request is a DISCONNECT. See “PAPL request and return codes” on page 138.)

    If there is DL/I activity at the time of the disconnect, and the disconnect is orderly (not immediate) DFHDBAT links to DFHDBSPX (the CICS-DBCTL suspend exit) to wait for all DL/I activity to complete. In this example, there was no DL/I activity at the time the disconnect was issued.
12. The DRA links to DFHDBSPX to cause the CICS task to wait while the DRA processes the disconnect request.
13. DBCTL return code (00000000). (See “Return codes in DBCTL” on page 137.)
14. Statistics for this session are recorded. (See “DBCTL statistics” on page 142.)
15. DFHDBDSC writes a record to the CICS global catalog, to indicate that CICS is no longer connected to DBCTL.
16. Phase 2 of disconnection begins.
17. DFHDBDI’s associated transaction, CDBD, runs and disables DFHDBAT to make it unavailable. (The transaction number changes from 00035 to 00032.)
18. Programs loaded at startup are disabled. This example shows DFHDBSPX. A complete trace should also include similar entries for other programs loaded at startup, as listed in “Connection to DBCTL” on page 123.
19. Message DFHDB8102 is issued confirming that disconnection from DBCTL is complete.

PSB schedule

This table shows an example of some CICS trace entries produced at PSB schedule time.

```
                                                .
                                                .
1    1CICS/ESA - AUXILIARY TRACE FROM 07/20/95 - APPLID CICSKPG1 - TIME OF FIRST ENTRY ON THIS PAGE 11:26:58.7144860002
                                                .
                                                .
      00039 1 AP 00E1 EIP   ENTRY CALLDLI                                          0004,00182718 ....,00004000 .. .
2,3   00039 1 AP 0328 DLI   ENTRY FUNCTION_CODE(PCB )   000C7526,TDLRA1
                                                .
                                                .
      00039 1 AP 0330 DLIDP ENTRY DBCTL
                                                .
                                                .
      00039 1 AP 2520 ERM   ENTRY APPLICATION-CALL-TO-TRUE(DBCTL   )
                                                .
                                                .
      00039 1 AP 2522 ERM   EVENT PASSING-CONTROL-TO-TRUE(DBCTL   )
      00039 1 AP 0310 DBAT  ENTRY APPLICATION           REQUEST
4,5   00039 1 AP 0311 DBAT  EVENT ABOUT-TO-INVOKE-DRA   FOR THREAD REQUEST , C3C9C3E2D2D7C7F1AB6538123994CA01,0301
6     00039 1 AP 0304 DBSPX EVENT ABOUT-TO-ISSUE-WAIT   FOR THREAD REQUEST
      00039 1 DS 0004 DSSR  ENTRY WAIT_MVS              DLSUSPND,DBCTL,0732001C,NO,OTHER_PRODUCT
      00039 1 DS 0005 DSSR  EXIT  WAIT_MVS/OK
      00039 1 AP 0305 DBSPX EVENT POSTED                FOR THREAD REQUEST
4,7   00039 1 AP 0312 DBAT  EVENT RECEIVES-CONTROL-FROM-DRA FOR THREAD REQUEST , C3C9C3E2D2D7C7F1AB6538123994CA01,00000000
      00039 1 AP 0313 DBAT  EXIT  DBAT-RESPONSE-CODE    (00000000)
      00039 1 AP 2523 ERM   EVENT REGAINING-CONTROL-FROM-TRUE(DBCTL   )
                                                .
                                                .
      00039 1 AP 2521 ERM   EXIT  APPLICATION-CALL-TO-TRUE(DBCTL   )
      00039 1 AP 0331 DLIDP EXIT  DBCTL
      00039 1 AP 0329 DLI   EXIT  IMS_PCB_FORMAT        0000,0000,PCB
8     00039 1 AP 00E1 EIP   EXIT  CALLDLI               OK                          00F4,00000000 ....,00004000 .. .
```

Figure 43. CICS trace entries produced for successful PSB schedule

Note:

1. Timestamp, as mentioned in "Connection to DBCTL" on page 123.
2. DL/I command or call type: PCB indicates a schedule request using the DL/I call interface.
3. PSB name (TDLRA1).
4. Recovery token (C3C9C3E2D2D7C7F1AB6538123994CA01).
5. The DRA is invoked for a thread request: 0301 is a PSB schedule request. (See "PAPL request and return codes" on page 138.)
6. DFHDBAT must wait, because the request has entered IMS code.
7. The DFHDBAT wait ends and DBCTL return code (00000000) is issued. The DBCTL return code is 00000000 because the PSB was successfully scheduled. See Figure 44 on page 130 for an example of the DBCTL return code in the case of a PSB scheduling failure. See "Return codes in DBCTL" on page 137 for an explanation of DBCTL return codes.
8. 00 in the UIBFCTR, and 00 in the UIBDLTR (underscored in this example) indicate that the PSB was scheduled successfully. See "PSB scheduling failure" for an example of the contents of these fields, PSB scheduling fails. See "Summary of DBCTL abends and return codes" on page 113 for information on the UIBFCTR and UIBDLTR.

PSB scheduling failure

An example of the trace entries produced if PSB scheduling fails.

```
1    1CICS/ESA - AUXILIARY TRACE FROM 07/20/95 - APPLID CICSKPG1 - TIME OF FIRST ENTRY ON THIS PAGE 11:26:58.7144860002
                                                  .
                                                  .
      00064 1 AP 00E1 EIP   ENTRY CALLDLI                                          0004,00182718 ....,00004000 .. .
2,3   00064 1 AP 0328 DLI   ENTRY FUNCTION_CODE(PCB )   000C8946,TXLRA1
                                                  .
                                                  .
      00064 1 AP 0330 DLIDP ENTRY DBCTL
                                                  .
                                                  .
      00064 1 AP 2520 ERM   ENTRY APPLICATION-CALL-TO-TRUE(DBCTL   )
                                                  .
                                                  .
      00064 1 AP 2522 ERM   EVENT PASSING-CONTROL-TO-TRUE(DBCTL   )
      00064 1 AP 0310 DBAT  ENTRY APPLICATION           REQUEST
4,5   00064 1 AP 0311 DBAT  EVENT ABOUT-TO-INVOKE-DRA   FOR THREAD REQUEST , C3C9C3E2D2D7C7F1AB654BD5E4F07E04,0301
6     00064 1 AP 0304 DBSPX EVENT ABOUT-TO-ISSUE-WAIT   FOR THREAD REQUEST
      00064 1 DS 0004 DSSR  ENTRY WAIT_MVS              DLSUSPND,DBCTL,0732001C,NO,OTHER_PRODUCT
      00064 1 DS 0005 DSSR  EXIT  WAIT_MVS/OK
      00064 1 AP 0305 DBSPX EVENT POSTED                FOR THREAD REQUEST
      00064 1 AP 0312 DBAT  EVENT RECEIVES-CONTROL-FROM-DRA FOR THREAD REQUEST , C3C9C3E2D2D7C7F1AB654BD5E4F07E04,880001AC
      00064 1 AP 0313 DBAT  EXIT  DBAT-RESPONSE-CODE    (00000000)
      00064 1 AP 2523 ERM   EVENT REGAINING-CONTROL-FROM-TRUE(DBCTL   )
      00064 1 AP 2521 ERM   EXIT  APPLICATION-CALL-TO-TRUE(DBCTL   )
      00064 1 ME 0301 MEME  ENTRY SEND_MESSAGE          1FAD,00051230 , 00000004,0011F5D0 , 00000005,0011F5D5 , 00000008,0011F3CC
      00064 1 ME 0501 MEIN  ENTRY INQUIRE_MESSAGE_DATA  86BB5AE0,DFHMET1E,1FAD,073017ED , 00000000 , 0000001C,07301787 , 00000000
                                                  .
                                                  .
7     00064 1 DU 0500 DUDT  ENTRY INQUIRE_SYSTEM_DUMPCODE DB8109
                                                  .
                                                  .
      00064 1 AP 0331 DLIDP EXIT  DBCTL
8     00064 1 AP 0329 DLI   EXIT  IMS_PCB_FORMAT        0805,0000,PCB
      00064 1 AP 00E1 EIP   EXIT  CALLDLI               OK                          00F4,00000000 ....,00004000 .. .
```

Figure 44. CICS trace entries produced for failed PSB schedule

Note:

1. Timestamp, as explained in "Connection to DBCTL" on page 123.
2. DL/I command or call: PCB indicates a schedule request using the DL/I call interface.
3. PSB name (TXLRA1).
4. Recovery token (C3C9C3E2D2D7C7F1AB654BD5E4F07E04).
5. The DRA is invoked for a thread request: 0301 is a PSB schedule request. (See "PAPL request and return codes" on page 138.)
6. The reason for the PSB scheduling failure is in the DBCTL return code (880001AC). In this case, it is X'1AC', indicating an IMS user abend U0428 (decimal), which was issued because the PSB was not defined to DBCTL.
7. Message DFHDB8109 is issued. It contains the IMS user abend, the recovery token, and the DBCTL ID. (For an example and explanation of how messages are displayed in abbreviated trace, see "Connection to DBCTL" on page 123.)
8. 0805 (underscored in this example) indicates that a PSB scheduling failure has occurred. 08 is in the UIBFCTR, and 05 in the UIBDLTR. (See "Summary of DBCTL abends and return codes" on page 113 for information on the UIBFCTR and UIBDLTR.)

CICS task issuing DL/I requests to be processed by DBCTL

An example of CICS trace entries produced when a DL/I request is issued.

For an example of trace entries produced by DBCTL for processing of a DL/I request, see "Trace entries produced by DBCTL" on page 132.

```
1    1CICS/ESA - AUXILIARY TRACE FROM 07/20/95 - APPLID CICSKPG1 - TIME OF FIRST ENTRY ON THIS PAGE 11:26:58.7144860002
      00040 1 AP 00E1 EIP   ENTRY CALLDLI                                              0004,00183718 ....,00004000 .. .
2,3   00040 1 AP 0328 DLI   ENTRY FUNCTION_CODE(GU  )   0001A8AC,DLIDBDR
      00040 1 AP 0330 DLIDP ENTRY DBCTL
      00040 1 AP 2520 ERM   ENTRY APPLICATION-CALL-TO-TRUE(DBCTL   )
      00040 1 AP 2522 ERM   EVENT PASSING-CONTROL-TO-TRUE(DBCTL   )
      00040 1 AP 0310 DBAT  ENTRY APPLICATION           REQUEST
4,5   00040 1 AP 0311 DBAT  EVENT ABOUT-TO-INVOKE-DRA   FOR THREAD REQUEST , C3C9C3E2D2D7C7F1AB653817A31F9000,0303
      00040 1 AP 0304 DBSPX EVENT ABOUT-TO-ISSUE-WAIT   FOR THREAD REQUEST
      00040 1 DS 0004 DSSR  ENTRY WAIT_MVS              DLSUSPND,DBCTL,0739501C,NO,OTHER_PRODUCT
      00041 1 DS 0005 DSSR  EXIT  WAIT_MVS/OK
      00041 1 AP 0305 DBSPX EVENT POSTED                FOR THREAD REQUEST
4,6   00041 1 AP 0312 DBAT  EVENT RECEIVES-CONTROL-FROM-DRA FOR THREAD REQUEST , C3C9C3E2D2D7C7F1AB653817A6C96600,00000000
      00041 1 AP 0313 DBAT  EXIT  DBAT-RESPONSE-CODE    (00000000)
      00041 1 AP 2523 ERM   EVENT REGAINING-CONTROL-FROM-TRUE(DBCTL   )
      00041 1 RM 0301 RMLN  ENTRY SET_LINK              01050000,073D69D4 , 00000000 , 00000008,NECESSARY,
      00041 1 RM 0302 RMLN  EXIT  SET_LINK/OK
      00041 1 AP 2521 ERM   EXIT  APPLICATION-CALL-TO-TRUE(DBCTL   )
      00041 1 AP 0331 DLIDP EXIT  DBCTL
7     00041 1 AP 0329 DLI   EXIT  IMS_PCB_FORMAT        0000,0000,PCB
      00041 1 AP 00E1 EIP   EXIT  CALLDLI               OK                              00F4,00000000 ....,0     0004000 .. .
```

Figure 45. CICS trace entries produced for a DL/I request

Note:

1. Timestamp, as mentioned in "Connection to DBCTL" on page 123.
2. DL/I command or call: GU indicates a GET UNIQUE request. (See "Comparing EXEC DLI commands and DL/I calls" on page 103 and "DL/I requests supported" on page 104.)
3. DBD name (DLIDBDR).
4. Recovery token (C3C9C3E2D2D7C7F1AB653817A31F9000). 3
5. The DRA is invoked for a thread request: 0303 is a DL/I request. (See "PAPL request and return codes" on page 138.)
6. DBCTL return code (00000000). (See "Return codes in DBCTL" on page 137.)
7. Status code in the DIBSTAT (underscored in this example) is 0000, indicating that the request was successful. See "Summary of DBCTL abends and return codes" on page 113 for the contents of DIBSTAT in the case of an unsuccessful request.

Thread termination

Example trace entries produced during PREPARE, COMMIT, and TERMINATE request processing.

See "Two-phase commit for DBCTL" on page 70 for a description of PREPARE and COMMIT request processing.

```
                                                .
                                                .
1    1CICS/ESA - AUXILIARY TRACE FROM 07/20/95 - APPLID CICSKPG1 - TIME OF FIRST ENTRY ON THIS PAGE 09:59:09.1299476250
2     00039 1 AP 2520 ERM   ENTRY SYNCPOINT-MANAGER-CALL-TO-TRUE(DBCTL   )
      00039 1 AP 2522 ERM   EVENT PASSING-CONTROL-TO-TRUE(DBCTL   )
      00039 1 AP 0310 DBAT  ENTRY SYNCPOINT-MANAGER     REQUEST
3,4   00039 1 AP 0311 DBAT  EVENT ABOUT-TO-INVOKE-DRA   FOR THREAD REQUEST , C3C9C3E2D2D7C7F1AB6538123994CA01,0304
      00039 1 AP 0304 DBSPX EVENT ABOUT-TO-ISSUE-WAIT   FOR THREAD REQUEST
      00039 1 DS 0004 DSSR  ENTRY WAIT_MVS              DLSUSPND,DBCTL,0732001C,NO,OTHER_PRODUCT
      00039 1 DS 0005 DSSR  EXIT  WAIT_MVS/OK
      00039 1 AP 0305 DBSPX EVENT POSTED                FOR THREAD REQUEST
3,5   00039 1 AP 0312 DBAT  EVENT RECEIVES-CONTROL-FROM-DRA FOR THREAD REQUEST , C3C9C3E2D2D7C7F1AB6538123994CA01,00000000
      00039 1 AP 0313 DBAT  EXIT  DBAT-RESPONSE-CODE    (00000004)
      00039 1 AP 2523 ERM   EVENT REGAINING-CONTROL-FROM-TRUE(DBCTL   )
      00039 1 AP 2521 ERM   EXIT  SYNCPOINT-MANAGER-CALL-TO-TRUE(DBCTL   )
                                                .
                                                .
      00039 1 AP 2520 ERM   ENTRY SYNCPOINT-MANAGER-CALL-TO-TRUE(DBCTL   )
      00039 1 AP 2522 ERM   EVENT PASSING-CONTROL-TO-TRUE(DBCTL   )
      00039 1 AP 0310 DBAT  ENTRY SYNCPOINT-MANAGER     REQUEST
3,6   00039 1 AP 0311 DBAT  EVENT ABOUT-TO-INVOKE-DRA   FOR THREAD REQUEST , C3C9C3E2D2D7C7F1AB6538123994CA01,0307
      00039 1 AP 0304 DBSPX EVENT ABOUT-TO-ISSUE-WAIT   FOR THREAD REQUEST
      00039 1 DS 0004 DSSR  ENTRY WAIT_MVS              DLSUSPND,DBCTL,0732001C,NO,OTHER_PRODUCT
                                                .
                                                .
      00039 1 DS 0005 DSSR  EXIT  WAIT_MVS/OK
      00039 1 AP 0305 DBSPX EVENT POSTED                FOR THREAD REQUEST
3,5   00039 1 AP 0312 DBAT  EVENT RECEIVES-CONTROL-FROM-DRA FOR THREAD REQUEST , C3C9C3E2D2D7C7F1AB6538123994CA01,00000000
      00039 1 MN 0201 MNMN  ENTRY MONITOR               1,DBCTL,7320090,100
      00039 1 MN 0202 MNMN  EXIT  MONITOR/OK
3,7   00039 1 AP 0311 DBAT  EVENT ABOUT-TO-INVOKE-DRA   FOR THREAD REQUEST , C3C9C3E2D2D7C7F1AB6538123994CA01,030F
      00039 1 AP 0304 DBSPX EVENT ABOUT-TO-ISSUE-WAIT   FOR THREAD REQUEST
      00039 1 DS 0004 DSSR  ENTRY WAIT_MVS              DLSUSPND,DBCTL,0732001C,NO,OTHER_PRODUCT
                                                .
                                                .
      00039 1 DS 0005 DSSR  EXIT  WAIT_MVS/OK
      00039 1 AP 0305 DBSPX EVENT POSTED                FOR THREAD REQUEST
3,5   00039 1 AP 0312 DBAT  EVENT RECEIVES-CONTROL-FROM-DRA FOR THREAD REQUEST , C3C9C3E2D2D7C7F1AB6538123994CA01,00000000
      00039 1 AP 2523 ERM   EVENT REGAINING-CONTROL-FROM-TRUE(DBCTL   )
8     00039 1 AP 2521 ERM   EXIT  SYNCPOINT-MANAGER-CALL-TO-TRUE(DBCTL   )
```

Figure 46. CICS trace entries produced during thread termination after DL/I request

Note:

1. Timestamp, as mentioned in "Connection to DBCTL" on page 123.
2. Enters syncpoint manager.
3. Recovery token (C3C9C3E2D2D7C7F1AB6538123994CA01).
4. The DRA is invoked for a thread request: 0304 is a PREPARE request. See "PAPL request and return codes" on page 138.
5. DBCTL return code (00000000), one for each of the requests PREPARE, COMMIT, and TERMINATE THREAD.
6. The DRA is invoked for a thread request: 0307 is a COMMIT request. See "PAPL request and return codes" on page 138.
7. The DRA is invoked for a thread request: 030F is a TERMINATE THREAD request. See "PAPL request and return codes" on page 138.
8. Leaves syncpoint manager. (See "Return codes in DBCTL" on page 137.)

Trace entries produced by DBCTL

In DBCTL, tracing is started by specifying an option in member DFSVSMxx in the IMS.PROCLIB (where xx is the suffix specified by VSPEC= in the DBCTL startup JCL).

See *IMS System Definition Reference* or *IMS Installation Volume 2: System Definition and Tailoring* for guidance on the DFSVSMxx member. Alternatively, you can start tracing dynamically with the /TRACE command. (See *IMS Operator's Reference* for guidance on the /TRACE command and its keywords.)

In DBCTL, you can start PI tracing in the DFSVSMxx member of the IMS.PROCLIB, as explained above. Alternatively, you can start PI tracing in DBCTL by issuing the command:

```
/TRACE SET ON PI
```

DBCTL produces an external trace when DL/I requests are issued to be processed by DBCTL. This trace corresponds to the CICS trace for a DL/I request being processed by DBCTL, as shown in Figure 45 on page 131. (DBCTL does not produce any external traces that correspond with the other CICS trace examples given.)

Figure 47 shows an example of the trace records produced when you use the DL/I trace table. To start the DL/I trace table, DLI=ON must have been specified in the DFSVSMxx member of IMS.PROCLIB. Specifying DLI=ON also enables program isolation and lock trace. For guidance on specifying DLI=ON, see *IMS System Definition Reference* or *IMS Installation Volume 2: System Definition and Tailoring*. Alternatively, you can start DL/I tracing dynamically using the /TRACE command, as follows:

```
/TRACE SET ON TABLE DL/I
```

For a more detailed example, see *IMS Operator's Reference* , example 8.

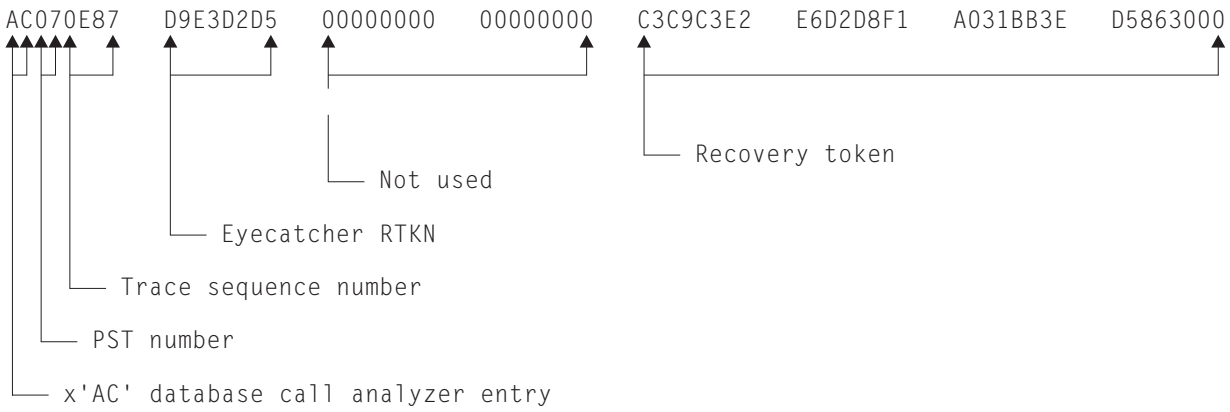


Figure 47. X'AC' trace entry

The DBCTL trace entry shown in Figure 47 includes:

- X'AC': the database call analyzer entry, which is only present for DBCTL.
- The partition specification table (PST) number. The PST number is equivalent to a particular DL/I thread number, as displayed using the /DISPLAY command, and can be used to find all DBCTL trace records for a particular thread. (For an example of a thread number being displayed, see "Purging a transaction that is using DBCTL" on page 60.)
- The trace sequence number.
- An "eyecatcher" recovery token. This is the actual characters "RTKN", used to draw attention to the recovery token in the same line, and is the same in every X'AC' entry.
- The recovery token that is passed from CICS via DFHDBAT.

You can print and format the above data using the IMS file select and formatting print utility, DFSERA10. You would typically print and format several log types, plus the X'AC' records to enable you to correlate the DBCTL activity with your CICS trace for a DL/I request.

Printing and formatting IMS X'67FA' log records

About this task

Figure 48 shows an example of JCL and DD statements that you can use to print and format IMS X'67FA' log records. For further examples, see *IMS Utilities Reference: Database* .

```
//LOGPRINT JOB 1,PGMERID,MSGCLASS=A,MSGLEVEL=(1,1),
//         CLASS=A
//ERA10     EXEC PGM=DFSERA10,REGION=4096K
//STEPLIB   DD DISP=SHR,DSN=IMS.RESLIB
//SYSPRINT  DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//LOGIN     DD DISP=SHR,DSN=IMS.SLDS.OLDS00
//SYSIN     DD *
CONTROL  CNTL  DDNAME=LOGIN
OPTION   PRINT OFFSET=5,FLDLEN=2,VALUE=67FA,COND=E,EXITR=DFSERA60
END
/*
//
```

Figure 48. Example JCL to print and format IMS '67FA' log records

The output should contain the following:

- The request type.
- The recovery token, plus an eyecatcher (GRTKN) to indicate presence of the recovery token, which includes the CICS APPLID.
- The database name.

See *IMS Utilities Reference: Database* for examples of formatted DL/I trace tables.

Dumps for CICS DBCTL

CICS, DBCTL, and the database resource adapter (DRA), produce a variety of dumps. Examining these dumps, particularly the CICS transaction or system dump, can help you determine whether a problem occurred in CICS or in DBCTL.

CICS transaction dump

This dump is produced whenever a CICS task terminates abnormally.

For a CICS-DBCTL task, that is, a task which has issued a DFHRMCAL request to DFHDBAT, this dump includes:

- The CICS-DBCTL global and task local areas
- DFHDBAT's global and task local areas
- PCBs

The recovery token for the task at the point of abnormal termination appears in the TCA (TCARTKN).

The EXEC CICS SET TRANDUMPCODE command and the CEMT SET TRANDUMPCODE transaction enable you to change some of the values recorded in entries in the transaction dump code table, to add new entries to the table, and to remove existing entries from the table. For example, you can specify an action for a particular CICS message, as mentioned in Figure 40 on page 124.

For information about transaction dump codes, and interpreting CICS dumps, see the *CICS Problem Determination Guide*.

CICS system dump

This dump is produced when a CEMT PERFORM DUMP|SNAP or an **EXEC CICS DUMP SYSTEM** command is issued, or when CICS abends.

CICS specifies all options when issuing this type of dump, for example, CSA and NUC. All MVS control blocks appear in this type of dump, including those corresponding to any subordinate TCBs. You can format and analyze this type of dump using the interactive problem control system (IPCS). For guidance on using IPCS, see *MVS Interactive Problem Control System: User's Guide*.

The **EXEC CICS SET SYSDUMPCODE** command and the CEMT SET SYSDUMPCODE transaction enable you to change some of the values recorded in entries in the transaction dump code table, to add new entries to the table, and to remove existing entries from the table. For example, you can specify an action for a particular CICS message, as mentioned in Figure 40 on page 124.

For information about system dump codes, and interpreting CICS dumps, see the *CICS Problem Determination Guide*.

Determining whether a problem is occurring in CICS or DBCTL

To help you determine whether a problem is occurring in DBCTL or CICS, examine the CICS transaction or system dump. These dumps include indications of the point at which DFHDBAT passes control to DBCTL and the point at which DBCTL returns control to DFHDBAT. Correlating this with the time at which the problem occurred should tell you whether it was in CICS or DBCTL.

Each page of auxiliary trace output also includes a timestamp, as mentioned in "Connection to DBCTL" on page 123. These timestamps should also help you correlate events in CICS with events in DBCTL.

DRA snap data set

The DRA's snap data set is dynamically allocated to the CICS address space when DBCTL is connected.

The SYSOUT class used is determined by a parameter in the DRA startup table. The DRA dumps its control blocks (those associated with its own work unit and that of DBCTL) to this data set whenever a high order bit is set in PAPLRETC. (The participant adapter parameter list (PAPL) is a part of the DRA. For guidance on the PAPL and its contents, see the appropriate *IMS Customization Guide*.) The high order bit is set on if a thread is terminating. It then closes the snap file. The recovery token appears in the dump produced.

What is provided in a CICS dump

When a transaction abends or requests a dump, the following areas are written to the CICS dump data set(s).

- The TCA representing the task.
- The CSA and CSA optional feature list (CSAOPFL) table. The CSAOPFL points to DFHDLPDS, the CICS-DL/I interface parameter block.
- The internal trace table, if CICS trace was active.
- Any areas acquired.

Dumps produced by the DRA

The DRA produces an SDUMP in these situations.

DBCTL creates an SDUMP containing diagnostic information for a DL/I request failure from CICS using the system dump data sets from the CICS job.

- If the DRA fails
- If a thread fails
- If DL/I set a high order bit in PAPLRETC for a thread request

  However, the DRA does not always take a dump if DL/I sets the high order bit in PAPLRETC. If it does not, it sets the second high order bit on to indicate this. For example:

  – If PAPLRETC is 1000 0000 3 2 4 0 0 0, a dump was taken

  – If PAPLRETC is 1000 1000 3 2 4 0 0 0, a dump was not taken

  (See "Return codes in DBCTL" on page 137, "Using return codes to find out what kind of dump has been produced" on page 138 and "PAPL request and return codes" on page 138 for information on interpreting these return codes.)

An SDUMP is created in a terminate address space request or a terminate thread request while running in DBCTL and under the DRA TCB.

An SDUMP contains:

- DBCTL address space
- DLISAS address space
- A storage list for the DRA area on the request
- Key 0 CSA storage for the request processing
- MVS storage blocks: address space control block (ASCB), TCB, and RBS for the failing DRA TCB
- The local system queue area (LSQA)

If the SDUMP request fails, a SNAP dump (which contains a subset of the information in an SDUMP) is produced instead. (See "Return codes in DBCTL" on page 137.) The SNAP contains the following subset of the information produced in an SDUMP:

- MVS storage blocks: address space control block (ASCB), TCB, and RBS for the failing DRA TCB
- A storage list for the DRA area on the request

Because the DRA runs in problem state, it cannot access other storage areas, such as CSA or DBCTL storage. This may mean that the SNAP does not contain enough information, and you may have to re-create the failure and use the DBCTL address space dump.

See *IMS Diagnosis Guide and Reference* for a further comparison of the information produced in SDUMPs and SNAP dumps, which you may find useful in diagnosis. The *IMS Diagnosis Guide and Reference* also contains information on the IMS offline dump formatter (ODF) which you can use to show the layout of IMS blocks referred to in these dumps.

Dumps produced by DBCTL

The formatted dump feature of IMS is available with DBCTL. This feature formats the system, database, and data communication areas of IMS.

It formats the control blocks and data areas in an IMS region.

See *IMS Diagnosis Guide and Reference* for guidance information on the areas that are dumped.

Control blocks generated by DBCTL have an "eyecatcher" for visual identification. For example:

- **SCD : system contents directory area
- **SSA : SAP and save area
- **DSP : dispatcher area.

The recovery token is included in dumps produced by DBCTL. Output is to the IMS log.

Messages for CICS DBCTL

DBCTL-related messages fall into these categories.

- Messages issued by the CDBC transaction and displayed on your screen. These messages relate to the end user's interaction with the transaction and they do not appear on CSMT. Any CDBC type messages issued from the initialization transaction, when it is running from the PLT during CICS startup, are issued as writes-to-operator (WTOs).
- Messages that appear on the status line of the CDBC and CDBI transaction screens.

CICS and IMS messages relating to CICS tasks that issue DL/I requests include the recovery token. See also "Dealing with messages from DBCTL and CICS" on page 63.

CICS messages relating to DBCTL begin with DFHDB81 or DFHDB82.

All DBCTL-related messages are routed to a separate destination called CDBC. If you prefer, you can direct them elsewhere (for example to CSMT).

You can suppress or reroute messages sent to transient data queues such as CDBC. You can reroute from CDBC to a list of consoles, from CDBC to a different transient data queue, or reroute console messages to CDBC. For programming information on coding the CICS-supplied user exit used to reroute messages and on the sample user exit provided to help you do so, see the *CICS Customization Guide*.

Messages produced with DBCTL dumps and traces are sent to the DBCTL master terminal operator. IMS messages begin with "DFS". See *IMS Messages and Codes* for guidance on interpreting, and responding to, IMS messages.

Return codes in DBCTL

When DBCTL responds to CICS with a return code, this can be an MVS system abend code, an IMS user abend code, or a DBCTL return code.

The return code includes an indicator to help you determine what kind of abend it is. The DBCTL return code (also known as the PAPLRETC) displayed in the CICS trace can contain:

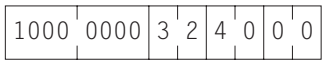
- An MVS system abend code
- A user abend code (also known as a pseudo abend code)

- A DBCTL return code (also known as a DBCTL DRA return code)

The return code is 4 bytes long and is in the following form:

If the top bit (bit 0 of the HH byte) is set:

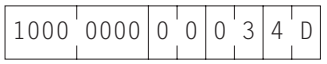
- either SSS is a nonzero hexadecimal return code, for example:




```
324 (hex) system abend code = 804 (decimal)
 MVS system abend
```

which indicates an MVS system abend code (as explained in *z/OS MVS System Codes*),

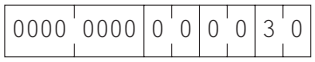
- or UUU is a nonzero hexadecimal, for example:




```
34D (hex) IMS user abend code = 845 (decimal)
 IMS user abend
```

which indicates a user abend code (as explained, for guidance, in the section on user abend codes in *IMS Messages and Codes*).

If the top bit (bit 0 of the HH byte) is ***not*** set, and the DBCTL return code in the CICS trace is nonzero, then UUU is a DBCTL nonzero return code, for example:




```
30 (hex) DBCTL return code = 48 (decimal) DBCTL
 return code
```

as explained, for guidance, in the DBCTL return codes section of *IMS Messages and Codes* .

Using return codes to find out what kind of dump has been produced

The top byte of the return codes indicates whether a dump has been produced and, if so, whether it is an SDUMP or a SNAP dump.

- X'80' means that an SDUMP or SNAP dump will be produced. (A SNAP dump is produced if the SDUMP request fails.)
- X'84' means that a SNAP dump only is produced.
- X'88' and X'00' both mean that neither an SDUMP nor a SNAP dump is produced.

See *IMS Messages and Codes* for guidance on interpreting IMS return codes and DBCTL return codes (also known as DRA return codes). Messages issued by CICS also distinguish the kind of return code you are receiving.

PAPL request and return codes

The trace examples given contain a number of 4-digit hexadecimal request codes issued by the participant adapter parameter list (PAPL). These request codes are a concatenation of a 2-digit PAPL function code and a 2-digit PAPL subfunction code. For further guidance on the contents of the PAPL, see the appropriate *IMS Customization Guide*.

Table 13 summarizes the PAPL request codes that are sent from CICS to the DRA, and are displayed in CICS trace output as 4-digit request codes. See "Trace entries produced by CICS" on page 122 for examples of traces containing these request codes.

Table 13. PAPL request codes

Event	Request code
Connection	0100
Disconnection	0400
Disconnection due to CICS failure	0404
PSB schedule	0301
DL/I request	0303
COMMIT request	0307
PREPARE request	0304
Single-phase SYNCPOINT request	030A
ABORT request	030D
Terminate thread	030F
COMMIT request during resynchronization	0201
ABORT request during resynchronization	0202
Lost because CICS was initial started before resynchronization	0203
DBCTL should not be indoubt	0204

Table 14 summarizes the PAPL return codes that are sent from the DRA to CICS. CICS intercepts these return codes and displays them as explanatory text in trace output.

Table 14. PAPL return codes

Event	Return code
Connection complete	0500
Identify failure	0501
Connection request (DRA INIT) canceled in reply to DFS690 message	0502
DBCTL has terminated abnormally	0503
The DRA has terminated abnormally	0504
/CHECKPOINT FREEZE or /CHECKPOINT PURGE command was issued to terminate DBCTL normally	0505

Using CICS EDF to debug application programs in DBCTL

You can use the CICS execution (command-level) diagnostic facility (EDF), with local and remote application programs that access databases controlled by DBCTL.

EDF supports the additional EXEC DLI commands and keywords that you can use with DBCTL, and the additions to the DL/I interface block (DIB) mentioned in "QUERY and REFRESH DBQUERY commands" on page 95.

However, a number of storage areas that resided in the CICS address space with local DL/I are outside the CICS address space with DBCTL. These areas include

the PDIR, DDIR, the PSB pool, and the DMB pool. You cannot access these areas using the WORKING STORAGE option of the CEDF transaction that invokes EDF. Instead, you use the DBCTL operator command /DISPLAY (with the keywords PSB, DBD, or POOL) to display the corresponding DBCTL information.

For information on using EDF, see the *CICS Application Programming Guide*.

Chapter 8. Statistics, monitoring, and performance for DBCTL

As with your CICS or IMS system, observing the performance of DBCTL involves collecting and interpreting data gathered by various CICS and IMS performance tools.

Note: In CICS and IMS, the term **statistics** means that data produced concerning timing and resources used by the system as a whole over a specified period. Additionally, in CICS, **monitoring** means data produced concerning timing and resources used by a task or a logical unit of work (UOW). IMS does not make this distinction: all data returned is referred to as statistics. Here, we use the terms statistics and monitoring in the CICS sense.

For information about CICS statistics and monitoring, see in the *CICS Performance Guide*.

For information about IMS performance and tuning, see *IMS System Administration Guide* or *IMS Administration Guide: System*.

Data available for a CICS-DBCTL system

As with your CICS or IMS system, observing the performance of DBCTL involves collecting and interpreting data gathered by various CICS and IMS performance tools.

The difference with DBCTL is that you need to keep an eye on events taking place in separate address spaces. Figure 49 on page 142 gives an overview of where DBCTL monitoring and statistics data is sent to and the tools you can use to produce output from this data. The data and tools mentioned are described in the sections that follow.

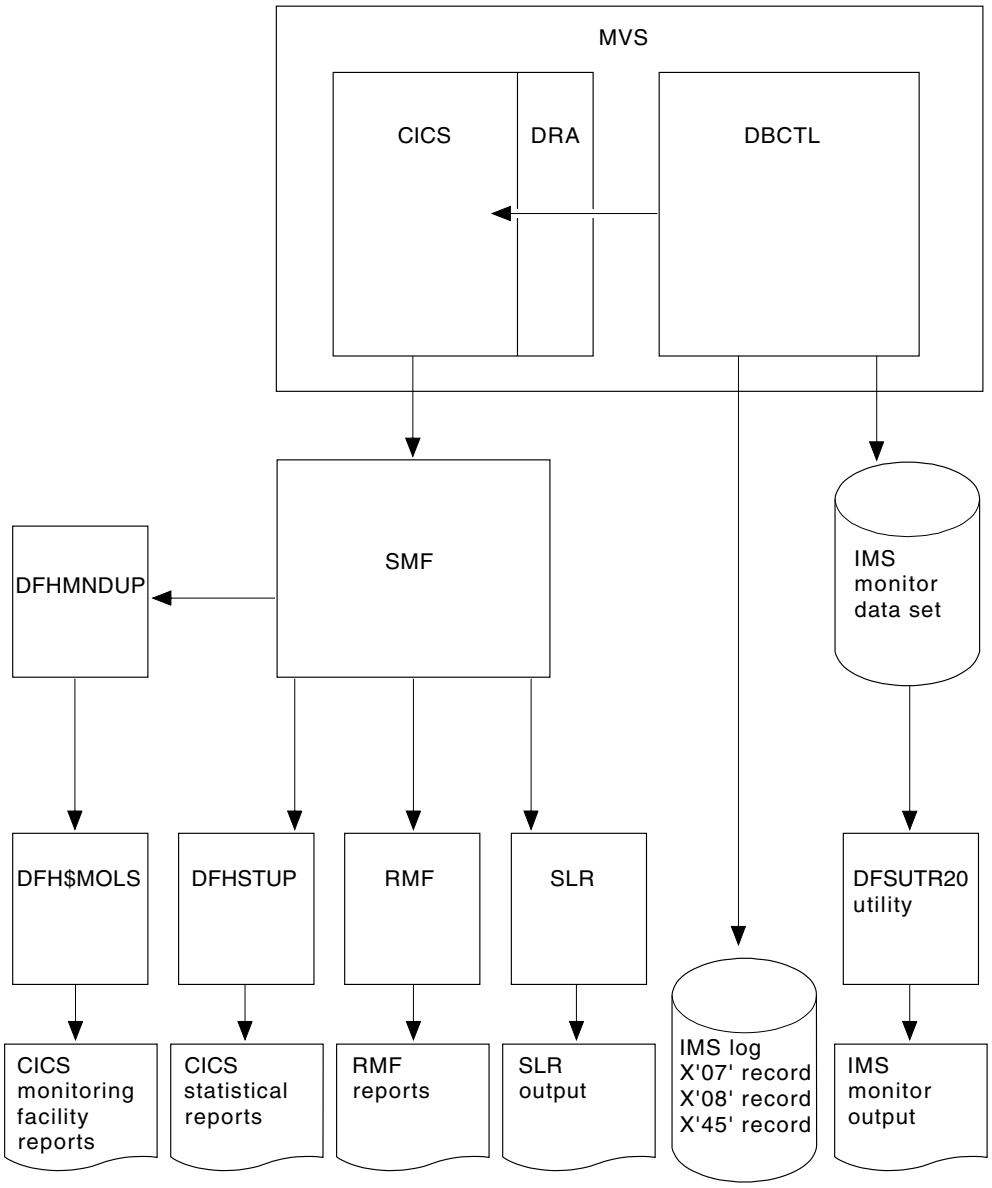


Figure 49. Overview of DBCTL statistics and monitoring data

DBCTL statistics

DBCTL supplies CICS with statistics information when CICS disconnects from DBCTL. These statistics are known as **unsolicited** statistics, because they are not produced as part of normal internal processing, but are produced as a z/OS UNIX System Services statistics record. The statistics are written to SMF regardless of the status of statistics recording.

CICS-DBCTL statistics are collected whenever DBCTL is disconnected as a result of:

- An orderly or immediate disconnection of DBCTL
- An orderly termination of CICS

CICS-DBCTL statistics are ***not*** collected if there is an immediate shutdown or abend of CICS.

When statistics are collected, the following actions occur:

1. The DRA returns statistics for the CICS-DBCTL session that has ended to DFHDBAT.
2. DFHDBAT invokes the CICS statistics exit for DBCTL statistics (DFHDBSTX).
3. DFHDBSTX invokes the CICS statistics domain.
4. The CICS statistics domain writes the statistics to the SMF data set.

CICS-DBCTL session statistics are contained in the DFHDBUDS DSECT, which you can generate from the copybook DFHDBUDS. DFHDBUDS includes the following information, which is returned from the DRA for that CICS session:

- DBCTL identifier for the CICS-DBCTL session (STATDBID).
- DBCTL recoverable service element (RSE) name (STARSEN). (For more information about RSEs, see Chapter 4, “Recovery and restart operations for DBCTL,” on page 65.)
- Time CICS connected to DBCTL (STACTIME).
- Time CICS disconnected from DBCTL (STADTIME).
- Minimum number of threads specified in the DRA startup table (STAMITHD).
- Maximum number of threads specified in the DRA startup table (STAMATHD).
- Number of times that the CICS-DBCTL session “collapsed” threads down to the minimum thread value specified in the DRA startup table (STANOMITHD).
- Number of times that the CICS-DBCTL session reached the maximum thread value specified in the DRA startup table (STANOMATHD).
- Elapsed time, expressed in hours, minutes, and seconds, for which the CICS-DBCTL session ran at the maximum thread value (STAELMAX).
- Peak number (also known as the “high watermark”) of thread TCBs created throughout the CICS-DBCTL session (STAHIWAT).
- Total number of times this CICS-DBCTL session successfully scheduled a PSB (STAPSBSU).

For information about DBCTL statistics see DBCTL session termination statisticshttp://publib.boulder.ibm.com/infocenter/cicsts/v4r2/topic/com.ibm.cics.ts.performance.doc/topics/dfht3_stats_dbctl.html.

To extract and print a report from these statistics, run the CICS-supplied statistics utility program (DFHSTUP), specifying the ***specific*** APPLID of the relevant CICS system. The output includes CICS-DBCTL session statistics provided DBCTL was connected to CICS when the statistics were collected. For information about other parameters needed to run DFHSTUP, and a sample job stream you can use, see . Figure 50 on page 144 shows an example of a report produced by running DFHSTUP.

```
Unsolicited Statistics Report      Collection Date-Time 09/16/93-15:16:18  Last Reset 15:06:46
-------------------------------------------------------------------------------------------
DBCTL SESSION TERMINATION STATISTICS
-----------------------------------
  CICS DBCTL Session Number           :             2
    DBCTL identifier                  :           SYS2
    DBCTL RSE name                    :      DBCTLSY2
    Time CICS connected to DBCTL      : 15:14:02.8506
    Time CICS disconnected from DBCTL : 15:16:18.3689
    Minimum number of threads         :             1
    Maximum number of threads         :             3
    Times minimum threads hit         :             1
    Times maximum threads hit         :             1
    Elapsed time at maximum threads   : 00:00:09.4371
    Peak number of thread TCBs        :             3
    Successful PSB schedules          :             9
```

Figure 50. Example of CICS-DBCTL session statistics output

Note: The statistics report produced by running DFHSTUP (shown in Figure 50) displays the times at which CICS connected to and disconnected from DBCTL in hours, minutes, and seconds (hhmmss) format in **local** time. The DBCTL z/OS UNIX System Services record mapped by the DFHDBUDS DSECT contains the connect and disconnect times as four 8-byte store clock (STCK) values. These values are: connect and disconnect time expressed in local time and connect and disconnect time in Greenwich Mean Time (GMT).

CICS statistics that contain the number of DL/I requests by type, issued against each DL/I database are ***not*** produced by CICS in the DBCTL environment. Instead, DBCTL produces this type of information. You can obtain DBCTL buffer pool utilization information from the DBCTL /DISPLAY command, or from the IMS log records of type X'45'.

Monitoring DBCTL: transaction level data

Monitoring data for DBCTL is passed to CICS and IMS components.

See the *CICS Operations and Utilities Guide* for help on switching monitoring on, and on printing and formatting the data.

DBCTL monitoring data returned to CICS

Monitoring data at the transaction level is passed back to CICS by DBCTL whenever a TERM request occurs, either explicitly, or implicitly at the end of task termination. The data is appended to the CICS monitoring facility performance record of the issuing task.

The data returned is as follows:

- PSB name.
- Elapsed wait time for pool space. In a PSB schedule, when the pool space is insufficient for PSB/DMB blocks, the schedule request is put on a wait queue. The total wait time for it is in this field.
- Elapsed wait time for intent conflict. In a PSB schedule, when an intent conflict is detected, the schedule request is put on a wait queue. The total wait time for it is in this field.
- Elapsed time for the schedule request.
- Elapsed wait time for database I/O.

- Elapsed wait time for locking. The total wait time to get the PI locks which are local segment level locks.
- Total number of database I/O counts.
- Number of DL/I requests for each of the following:
  - Get unique
  - Get next
  - Get next within parent
  - Get hold unique
  - Get hold next
  - Get hold next within parent
  - Insert requests
  - Delete requests
  - Replace requests
- Total number of DL/I database requests.
- Number of test enqueues.
- Number of times requesting the PI locks on segments.
- Number of waits on test enqueues.
- Number of times requesting the PI locks on segments.
- Number of dequeues.
- Number of times PI locks are released.
- Number of update enqueues.
- Number of times the update locks are not available for a request and requires a wait.
- Number of update dequeues.
- Number of times requesting the exclusive lock.
- Number of waits on exclusive enqueues.
- Number of times the exclusive locks are released.
- Number of exclusive dequeues.
- Number of times the exclusive locks are released.
- DEDB statistics:
  - Number of DEDB requests
  - Number of DEDB I/Os
  - Number of overflow buffers used
  - Number of waits for DEDB buffer
  - Number of unit of work contentions
- Date of schedule start.
- Time of schedule start.
- Date of schedule end.
- Time of schedule end.
- Elapsed UOW CPUTIME for DRA thread (see note below).

Note: The elapsed CPUTIME field was introduced by IMS APAR PL83370. The CPUTIME represents the time spent in the DRA Thread TCB from the time the PSB is scheduled, to the time the PSB is terminated. CICS always terminates the PSB at the end of the Unit of work (UOW). The CPUTIME does not include any time spent in the DBCTL region.

Calculating CICS and IMS processor times for IMS Version 12 or later

When CICS is connected to IMS Version 12 or later, and is using the open transaction environment (OTE), the CICS-DBCTL database adapter transformer DFHDBAT, uses CICS-managed L8 open TCBs rather than CICS IMS subtask TCBs. This means that the CICS monitoring facility can measure activity that was previously only reported in the IMS data that was returned whenever a TERM request occurred. For example, CICS can now measure the processor time consumed on the IMS thread. When CICS is using L8 open TCBs, the CPU time reported for these TCBs by the CICS monitoring facility includes the IMS elapsed UOW CPUTIME for the DRA thread.

When CICS is connected to IMS Version 12 or later, do not add the processor time from the CICS records (SMF type 110 records) and the IMS elapsed UOW CPUTIME when calculating the total processor time for a single transaction, because the IMS 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, H8, J8, and L8 mode TCBs.

In the OTE, the CICS L8 task processor time can also include the cost of creating an IMS DRA thread.

Also take the capture ratio for CICS and IMS into account. Capture ratio is the ratio of reported CPU time to total used CPU time. For more information, see the *z/OS Resource Measurement Facility Performance Management Guide*.

Obtaining DBCTL monitoring data sent to CICS

DBCTL supplies CICS with monitoring data, which is then output to the CICS monitoring domain in the following cases.

- When CICS receives the response to a PSB schedule request from DBCTL, it checks whether this task has already been scheduled successfully to DBCTL. If it has, CICS forces the monitoring data from the previous PSB schedule out; that is, it writes the performance class record for the task and resumes monitoring that task. If it has not been scheduled before, no monitoring processing is done.
- When CICS receives a response from the DBCTL as a result of a COMMIT or ABORT request, CICS outputs the monitoring data, but does not write it.
- In the case of the final PSB schedule for a task, the monitoring data is automatically written at the end of a task.

To obtain the monitoring data that DBCTL returns to CICS, code two additional event monitoring points (EMPs) in your CICS monitoring control table (MCT). DBCTL EMPs can be found in CICSTS42.CICS.SDFHSAMP member DFH$MCTD.

For programming information on EMPs and CICS monitoring, see the *CICS Performance Guide*.

When you have obtained the monitoring data, you can use monitoring tools such as the CICS monitoring facility and the Service Level Reporter (SLR) with the data supplied to tune your CICS-DBCTL environment.

Service Level Reporter (SLR)

Service Level Reporter (SLR) is an IBM database and reporting program that collects and analyzes data from CICS and other IBM systems and products.

SLR collects CICS data from CICS monitoring records and from a subset of CICS statistics on the SMF log data set. It then analyzes the data, summarizes the results, and stores the data in the SLR database.

The DBCTL data in the CICS monitoring records is output as one 256-byte block, and is written by the EMP DBCTL.2, as defined in CICSTS42.CICS.SDFHSAMP member DFH$MCTD. The DSECT for this 256-byte block of data is mapped by the DFSDSTA macro in the IMS GENLIBs. The SLR tables CICSTRANSLOG and CICSTRANSNUM contain the fields in this block.

During SLR installation, you must specify whether you want DBCTL data. For guidance on the format of this data, see the description of the DFSDSTA macro in the appropriate *IMS Customization Guide*. For information on SLR data in CICS and IMS, see *IMS System Administration Guide* or *IMS Administration Guide: System*. For help on using SLR, including examples of SLR reports and how to make use of them, see *Service Level Reporter General Information* .

Note: There is a follow-on product to SLR, called Performance Reporter for MVS, which has DB2 as a prerequisite. This product includes the functions that are carried out by SLR. *Performance Reporter for MVS CICS Performance Feature Guide and Reference*, SH19-6820, describes the way this product works with CICS.

IMS monitor reports with DBCTL

A summary of the DBCTL-related data in IMS monitor reports. This information also applies if your CICS system is connected to an IMS DM/TM system to obtain DBCTL support.

IMS monitor reports that apply to DBCTL

- Call summary
- Program I/O
- DB buffer pool
- VSAM buffer pool
- Program summary

Note: In a DBCTL environment, interpret the terms “program” and “transaction” in these reports as “PSB” and “PSB scheduling”, respectively.

IMS monitor reports that apply partially to DBCTL

- Region summary
- Region IWAIT

  (An IWAIT occurs when a DBCTL request causes I/O activity. IWAIT time denotes the time DBCTL spends waiting for IMS resources, in addition to the number of I/Os.)
- Any other region-based reports.

Note: In a DBCTL environment, interpret the term “region” in these reports as the representation of a CICS thread or a BMP region in DBCTL, but beware that a DBCTL region can represent different CICS threads or BMP regions during a monitor run.

IMS monitor reports that do not apply to DBCTL

The following reports, related to transaction management and communication, do not apply to DBCTL, and either do not appear, or are shown as headings without any data:

- Communication wait
- Communication summary
- Line functions
- Message format buffer pool
- Message queue pool
- MSC queuing summary
- MSC summaries
- MSC traffic

Data contained in relevant IMS monitor reports

This topic shows you what data you can find in the IMS monitor reports that apply to DBCTL.

General wait time events

All threads built for a CICS system have the same job name as that CICS system. They are shown in the jobnames for regions in the "General reports".

General reports

The "general reports" include the "Regions and jobname" report and the "Region summary report".

Regions and jobname report

Within a trace interval, a thread can be assigned to multiple CICS systems but it can only be assigned to one CICS at any one time.

Depending on the number of CICS systems connected to DBCTL, the regions and jobname report can show:

- One region with only one jobname.
- One region with multiple jobnames.
- Multiple regions with multiple jobnames where some regions have the same jobname, and some have multiple jobnames.
- Multiple regions with only one jobname.

Any monitor report for a ***region*** is a summary for all connected CICS systems that a ***thread*** has served during the trace interval. For example, the elapsed time of schedule end to first call means the sum of this elapsed time for all CICS systems that a thread has been assigned to during the trace interval.

Depending on the workload of a CICS system, a trace interval may be a relatively short period of time, and thread switching between depending regions may not occur very often. However, the more the workload fluctuates, the more frequently threads are likely to be assigned among connected CICS systems.

Region summary and transaction queuing report

A region summary report and a transaction queuing report can be used to show you the information about DBCTL.

Region summary report

A region summary report can include the following information about DBCTL: scheduling and termination, schedule to first call, elapsed execution, region occupancy, and DL/I calls.

- **Scheduling and termination**, including:
  - The time from PSB schedule request being received by DBCTL to when the request is completed by DBCTL. This includes the time spent by DBCTL allocating IMS resources and does not include any schedule time spent in CICS or being processed by the DRA.
  - The time from when a PSB unschedule request is received by DBCTL to when the request is completed by DBCTL. This request could be an unschedule PSB request, or a request embedded in any synchronization type terminate request, or a terminate thread request.
- **Schedule to first call** is the time from when DBCTL completed the PSB schedule to when DBCTL received the first DL/I request. This time includes all time spent processing in CICS, including application program, CICS itself, and DRA processing. (Because CICS is the transaction manager, how and when its own applications are loaded or scheduled cannot be interpreted by DBCTL in the IMS monitor reports.)
- **Elapsed execution** is the time between the completion of the DBCTL PSB schedule request and when DBCTL receives the PSB unschedule request. It indicates the amount of time IMS resources were allocated to a CICS thread.
- **Region occupancy** is the ratio of the elapsed time when a thread is active (that is, with IMS resources allocated) to the trace interval.
- **DL/I calls** is the time between DBCTL receiving the DL/I request and the request being completed in DBCTL.

Program summary

DBCTL does not process any messages. For the purpose of using the DC monitor report, it counts each PSB schedule as one message dequeued. Because DBCTL is not the transaction manager, it must assume a one-to-one relation between a CICS transaction and a PSB schedule. This relationship is shown in program summary, where the number of transactions dequeued is the same as the number of scheduled requests. “Per transaction” means requests per schedule, and “elapsed time per transaction” means elapsed time per schedule.

Run profile

In run profile, the number of messages dequeued means the number of scheduled PSBs and transactions per second means PSB schedules per second.

Transaction queuing report

The transaction queuing report can include a list of transactions for DBCTL. Each transaction name is an 8-byte transaction ID specified by CICS on the schedule request. A transaction ID from CICS consists of a 4 byte CICS transaction name, plus a 4 byte CICS identifier. If CICS does not specify a transaction ID, DBCTL takes the CICS region ID, obtained at connection time. In this report, for DBCTL,

the transaction “number dequeued” means number of PSB schedules. The “on queue when scheduled” in this report is always zero because the IMS message queues do not apply to DBCTL.

For examples of IMS monitor reports and detailed guidance on interpreting their contents, see *IMS Utilities Reference: Database*.

Using the IMS monitor

DBCTL enables CICS users who do not have an IMS/VS DB/DC or IMS/DM/TM system to use the IMS monitor online. The IMS monitor is the main tool provided by IMS for monitoring. It collects data from the system while it is running. It formats and records significant events during execution, and is useful in tuning constrained systems.

Monitoring data is written to a separate data set or tape defined by the IMSMON DD statement in the DBCTL JCL. To define this data set or tape and to run the IMS monitor with DBCTL, add an IMSMON DD statement to your DBCTL JCL. For further guidance on doing so, see *IMS System Definition Reference* or *IMS Installation Volume 2: System Definition and Tailoring*.

To allocate an IMSMON data set, use the IEFBR14 utility to allocate a data set without any DCB parameters; for example:

```
//ALLOC     EXEC PGM IEFBR14
//IMSMON DD DISP=(NEW,CATLG),UNIT=3380,VOL=SER=xxxxxx,SPACE=(CYL,(5,5))
```

You can start and stop the IMS monitor dynamically using the /TRACE command with the MON keyword. For example:

```
/TRACE SET ON MON ALL
```

gives you all the activity that the monitor collects. For guidance on using the /TRACE command and its keywords more selectively, see *IMS Operator's Reference*.

The IMS monitor has two phases:

- During the first phase, the monitor programs collect the data and store it on either disk or tape.
- During the second, the data is retrieved from the data set, and is organized and printed.

The data collected by the monitor (also known as DFSMNTR0) is organized and printed by the IMS monitor report print program, DFSUTR20. See *IMS Utilities Reference: Database* for guidance on using the IMS monitor report print utility, DFSUTR20, and for information about using the IMS monitor to identify constraints.

DBCTL data returned to IMS log

In addition to the information returned to the monitor, IMS writes monitoring information to the log records. This information is always recorded; you do not have to request it.

For further information about the data returned to the monitor see “IMS monitor reports with DBCTL” on page 147.

IMS appends the following information to the X'08' log records during ***scheduling***.

- Total elapsed wait time due to intent conflict

- Total elapsed wait time due to pool space not being available
- Total elapsed time for a schedule request

IMS appends the following information to the X'07' log records at PSB ***termination***:

- Total number of databases used involved in I/O
- Total number of DL/I database requests
- Total elapsed wait time due to databases involved in I/O
- Total elapsed wait time due to locking
- Total number of gets
- Total number of inserts
- Total number of replace
- Total number of deletes

Program isolation trace

For full function DL/I databases, you can use the program isolation (PI) trace to get records that indicate queueing activity taking place for program isolation. The PI trace records are written to the IMS log. You can then print them using the IMS file select and formatting utility. See *IMS System Administration Guide* for further guidance on using PI trace.

DL/I trace

For full function databases, you can use DL/I trace with DBCTL by enabling the DL/I trace table in the DFSVSMxx member or by issuing the /TRACE command.

The /TRACE command is described in "Controlling tracing of DBCTL events" on page 55. Using the /TRACE command enables you to turn DL/I trace on and off while the system is running. Output is to the IMS log as type X'67FA' records. See *IMS Diagnosis Guide and Reference* for guidance on using DL/I trace for diagnosis, *IMS Operator's Reference* for guidance on the commands needed to invoke it, and *IMS Utilities Reference: Database* for guidance on printing its output.

Using the IMS log statistics utilities

You can use these IMS log statistics utilities to process the information from the IMS log. See "DBCTL data returned to IMS log" on page 150 for a list of the data returned to the IMS log.

- File select and formatting print utility, DFSERA10, formats, and prints selected records from the IMS log data set. The active OLDS must have been archived before you can access the log data. You normally specify the SLDS to DFSERA10. You can also use DFSERA10 with the program isolation trace record format and print module, DFSERA40, to format PI trace.
- DEDB log analysis utility, DBFULTA0, prepares statistical reports for DEDBs based on data recorded on the IMS system log.
- IMS program isolation trace report utility, DFSPIRP0. If you use program isolation (PI), you can use DFSPIRP0 with the IMS log to obtain information about deadlocked tasks. DFSPIRP0 prints a report that shows only those enqueue requests that required a wait because the resource was not immediately available.

See *IMS Utilities Reference: Database* for guidance on using these utilities.

Trace facilities

CICS trace facilities are intended primarily as debugging tools. However, because they record all requests for CICS, you can use them to analyze the performance of individual transactions.

For information about trace entries produced in a DBCTL environment see Chapter 7, "Problem determination for DBCTL," on page 119. For information about specifying CICS trace parameters see the *CICS Problem Determination Guide*

CICS auxiliary trace facility

The CICS auxiliary trace facility enables you to record trace entries on a separate data set to be analyzed later. Trace entries are time-stamped and they can provide detailed information for analyzing constraints or other problems that can occur while CICS is running. For examples of CICS auxiliary trace output, see "Trace entries produced by CICS" on page 122.

However, consider carefully how often you must use CICS auxiliary trace because it generates a large volume of entries, which means that there might be a considerable overhead if you run it all the time. Also, you might find it difficult to make effective use of too large a volume of such data.

Additional performance tools

The following are additional performance tools that you may want to consider using with DBCTL if you already have them or are considering adding them to your system.

Generalized trace facility (GTF)

If you use the IRLM as your locking manager, you can use the generalized trace facility (GTF) to provide a trace of its activity. It traces request handler request completions, the PTB input/output buffers, and statistical data relevant to the IRLM.

You can print the records GTF produces offline. Output is collected in a data set specified by its user in the GTF job. For guidance on using GTF, which you may find of use in debugging, see *IMS Diagnosis Guide and Reference* .

Resource Measurement Facility (RMF)

The Resource Measurement Facility (RMF™) is a measurement tool designed to meet the needs of performance management in the large systems environment that MVS supports.

Its primary purpose is to reduce the amount of system programmer time and expertise required to identify and to diagnose system tuning problems. It is designed to monitor selected areas of system activity and present the data collected in the form of SMF records and/or formatted reports. Display reports are also available for some system activities. For more details, see the *CICS Performance Guide*, and *Resource Measurement Facility User's Guide*.

Tuning a CICS-DBCTL system

You can tune your CICS-DBCTL setup to make efficient use of resources to help you reach performance objectives.

Performance parameters in CICS

System design considerations for CICS with DBCTL are similar to the design considerations that applied to local DL/I. For example, do not allow excessive database accesses or updates in a single UOW.

However, there are some differences.

The fact that DBCTL is structured to have one TCB per thread is an additional consideration for CICS. This allows more concurrent processing, but you must be aware of the need to specify minimum and maximum numbers of threads that are consistent with the needs of your system. For more information, see "Specifying numbers of threads" on page 154.

The storage specified in CICS system initialization parameters DSALIM and EDSALIM is used for different resources in a CICS-DBCTL environment. DSALIM is used to specify the upper limit of the total amount of storage within which CICS can allocate the individual DSAs **below** the 16 MB line. EDSALIM is used to specify the upper limit of the total amount of storage within which CICS can allocate the individual EDSAs **above** the 16 MB line. Local uses DSA for PSB and DMB pools, but with DBCTL, these blocks are stored outside CICS. Instead, you need to allow for the storage DBCTL needs in CICS for DRA code when specifying DSALIM and EDSALIM. This storage is allocated in the CICS region, but not from DSA or EDSA storage. See the the *CICS Performance Guide* for information about specifying DSALIM and EDSALIM, and *IMS System Administration Guide* or *IMS Administration Guide: System* for guidance on DBCTL storage estimates.

Using single-phase commit

CICS can use single-phase commit instead of two-phase commit when, for a particular UOW, DBCTL is the only recoverable resource used. Using single-phase commit in these circumstances improves CICS performance with DBCTL by eliminating unnecessary logging, cutting restart time, decreasing transaction cost, and improving response time in both CICS and DBCTL. For information on using single-phase commit, see the *CICS Customization Guide*.

Performance parameters in IMS

From an IMS point of view, tuning DBCTL is much like tuning an IMS system.

Additional considerations are DRA threads, described in "Specifying numbers of threads" on page 154, and DEDBs, described in "DEDB performance and tuning considerations" on page 155.

Response time: assigning job dispatching priorities

To minimize response times, assign a higher dispatching priority to the CICS address space than to the DBCTL address spaces (DBCTL, DLISAS, DBRC).

Although CICS can be regarded as a "front end" to DBCTL, you must be aware that CICS also has to manage the network and the application environment for non-DLI transactions such as DB2 or VSAM. This means that CICS has different CPU requirements from other front ends to DBCTL such as a BMP or an MPP. For example, when a CICS transaction is waiting for a response to a DBCTL request, CICS dispatches other CICS transactions.

If IRLM is assigned a priority of *n*, CICS should have a priority of *n-1*, DBRC a priority of *n-2*, and DBCTL and DLISAS a priority of *n-3*.

For further guidance on assigning priorities, see *IMS System Administration Guide* or *IMS Administration Guide: System*.

Specifying numbers of threads

The DRA startup parameters, MINTHRD and MAXTHRD, specify the minimum and maximum numbers of threads that can process DBCTL DL/I or DEDB requests. The MINTHRD and MAXTHRD parameters are specified in the DRA startup table (DFSPZP).

See "Defining the IMS DRA startup parameter table" on page 27 for more information on DRA startup parameters.

The IMS system generation parameter, MAXREGN, specifies the number of regions (or threads), to be allocated at startup, that DBCTL can handle for all connected CICS systems and BMPs. The number can increase dynamically, to a limit of 999, as required. See "Generating DBCTL" on page 17 for information on system generation parameters.

The number you specify for MAXREGN should be no less than the sum of the MINTHRD parameters specified for active CICS systems, and for BMPs.

In Figure 51, the following threads are in use: one from BMPA, one from BMPB, five from CICSA and three from CICSB, making a total of 10 threads. A MAXREGN of 10 has therefore been specified for DBCTLA.

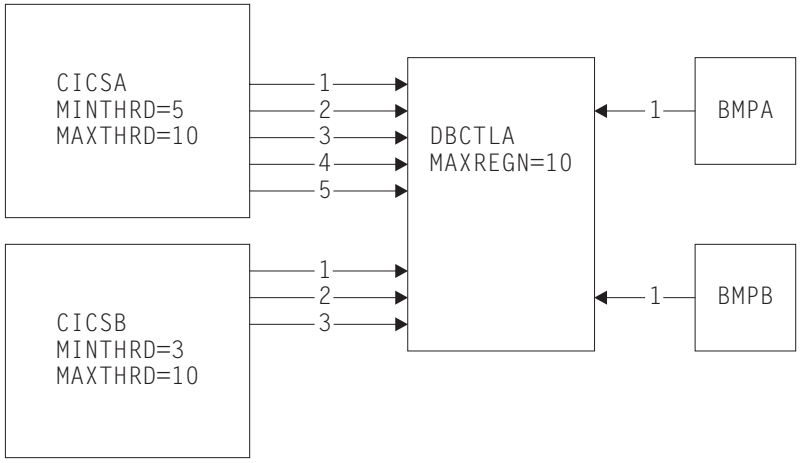


Figure 51. Interaction of MAXREGN, MINTHRD, and MAXTHRD

MAXTHRD can be used in DBCTL systems to ensure that, at peak loads, additional threads can be built in addition to those already allocated as a result of MINTHRD, thus avoiding waiting for threads. The maximum number of threads you can specify in DBCTL is 999. The default is 1 or the number defined by MINTHRD, whichever is the highest. MAXTHRD controls the maximum number of tasks for which this CICS system can have PSBs scheduled in DBCTL. Any requests to schedule a PSB when the MAXTHRD limit is reached is queued by the DRA. One thread is equivalent to one MVS TCB, thus giving more concurrency on multiprocessors. There is a storage allocation of about 9 KB per thread in the local system queue area (LSQA) below the 16 MB line. Because these threads are available for the duration of the DBCTL connection, there is no pathlength overhead for collapsing and reallocating thread related storage, and throughput should, therefore, be faster. The number of threads that you specify must be large

enough for your system's needs, but if you specify a number that exceeds those needs, this will have an adverse effect on the performance of the DRA. If you specify a minimum thread value that is higher than your system's actual minimum activity, this will tie up threads unnecessarily, preventing DBCTL from allocating them to other CICS systems or BMPs. If you specify a minimum thread value that is too low, this can also affect performance; if the level of thread activity falls, this could cause the DRA to release threads down to the minimum value. These threads would then have to be reestablished if the thread requests increased again.

The number you specify for MAXTHRD should reflect what you consider to be the peak load of DBCTL threads needed. The number of threads you specify will affect performance. The larger the number you have preallocated, the more storage is needed. However, if threads are preallocated, the time needed to allocate them on demand is saved, thus improving response time and throughput. So, if your system is storage constrained, specify a lower value for MINTHRD, and use MAXTHRD as a "safety valve". If response time and throughput are more important than storage requirements, specify a higher number for MINTHRD so that more threads are ready to be used.

After the MINTHRD limit is exceeded, threads continue to be built up to the MAXTHRD limit but, because each thread's control blocks are allocated during PSB scheduling, the pathlength is greater for the tasks running after the MINTHRD limit has been reached.

Also bear DBCTL thread activity in mind when specifying the MXT system initialization parameter. You use MXT to specify the maximum number of tasks that CICS will allow to exist at any time. With DBCTL, MXT should be enough to allow for the number specified in MINTHRD, plus the number you need for "standard" CICS tasks. With DB2, there is no minimum number of threads. See the *CICS Performance Guide* for general help on MXT.

To help you decide on the optimum values for minimum and maximum numbers of DBCTL threads, monitor thread usage and IMS task throughput (to see if tasks are being delayed), and IMS I/O rates. For details of thread statistics produced, including maximum and minimum thread usage, see "DBCTL statistics" on page 142. See "DBCTL data returned to IMS log" on page 150 for details of data produced for monitoring IMS I/O rates. You can also use CICS auxiliary trace to check for queueing for threads and PSBs.

DEDB performance and tuning considerations

If you use DEDBs, you must define the characteristics and usage of the IMS DEDB buffer pool. You do this by specifying parameters both in the CICS region and the IMS (DBCTL) region. The DBCTL DEDB parameters are useful when tuning a CICS/DBCTL DEDB fastpath environment. DBBF and DBFX are parameters defined during DBCTL system generation or at DBCTL initialization. CNBA, FPBUF, and FPBOF are defined in the DRA startup table (DFSPZP).

All parameters that you need to during IMS system definition or execution, including DRA startup parameters, are described in "Defining the IMS DRA startup parameter table" on page 27) .

The main concerns in defining DEDB buffer pools are the total number of buffers in the IMS region, and how they are shared by CICS threads. You use the following IMS FPCTRL parameters to define the number of buffers:

- DBBF: total number of buffers
- DBFX: number of buffers used exclusively by the DEDB system.

The number remaining when you subtract the value specified for DBFX from the value specified for DBBF is the number of buffers available for the needs of CICS threads; for this example, It is a fixed number is assumed for DBFX. DBBF must, therefore, be large enough to accommodate all batch message processing programs (BMPs) and CICS systems that you want to connect to this DBCTL system.

When a CICS thread connects to IMS, its DEDB buffer requirements are specified using a normal buffer allocation (NBA) parameter. For a CICS system, there are two NBA parameters in the DRA startup table:

1. CNBA buffers needed for the CICS system. This is taken from the total specified in DBBF.
2. FPBUF buffers to be given to each CICS thread. This is taken from the number specified in CNBA. FPBUF is used for each thread that requests DEDB resources, and so should be large enough to handle the requirements of any application that can run in the CICS system.

A CICS system might fail to connect to DBCTL if its CNBA value is more than that available from DBBF. An application might receive schedule failure if the FPBUF value is more than that available from CNBA. The FPBUF value is used when an application tries to schedule a PSB that contains DEDBs.

When a CICS system has successfully connected to DBCTL, and the application has successfully scheduled a PSB containing DEDBs, the DRA startup parameter FPBOF becomes relevant. FPBOF specifies the number of overflow buffers each thread will get if it exceeds FPBUF. These buffers are not taken from CNBA. Instead, they are buffers that are *serially* shared by all CICS applications or other dependent regions that are currently exceeding their NBA allocation.

Because overflow buffer allocation (OBA) usage is serialized, thread performance can be affected by NBA and OBA specifications. If FPBUF is too small, more applications need to use OBA, which may cause delays due to contention. If both NBA and OBA are too small, the application fails. If FPBUF is too large, this affects the number of threads that can concurrently access DEDB resources, and increases the number of schedule failures.

In a CICS-DBCTL environment, the main performance concern is the trade-off between speed and concurrent access. The size of this trade-off is dictated by the kind of applications you are running in the CICS system. If the applications have approximately the same NBA requirements, there is no trade-off. You can specify a FPBUF large enough to never need OBA. This speeds access and there is no waste of buffers in CNBA, thus enabling a larger number of concurrent threads using DEDBs. The more the buffer requirements of your applications vary, the greater the trade-off. If you want to maintain speed of access (because OBAs are not being used) but decrease concurrent access, you should increase the value of FPBUF. If you prefer to maintain concurrent access, do not increase the value of FPBUF. However, speed of access will decrease because this and possibly other threads will need to use the OBA function.

For information on specifying the parameters CNBA, FPBOF, and FPBUF, see "Defining the IMS DRA startup parameter table" on page 27. For further guidance on DEDB buffer specification and tuning, see sections on DEDBs in *IMS Administration Guide: Database Manager* and *IMS Administration Guide: System*.

Monitoring data at the transaction level is returned to CICS by DBCTL at schedule end and transaction termination. This data includes DEDB statistics. To obtain the monitoring data, two event monitoring points (EMPs) must be added to your CICS monitoring control table (MCT).

Using DEDBs

Using DEDBs can provide performance improvements in a number of areas, including a reduction in path length, parallel processing capability, less I/O processing, and a reduced logging overhead.

- Reduced path length
  - DEDBs use Media Manager for more efficient control interval (CI) processing, which can reduce pathlength.
  - DEDBs have their own resource manager, which means:
    - Less interaction with whichever lock manager you are using (PI or the IRLM), provided you are not using block level sharing.
    - Simplified buffer handling (and reduced pathlength) because DEDBs have their own buffer pool.
- Parallel processing

  DEDB writes are not done during the life of the transactions but are kept in buffers. Actual update operations are delayed until a synchronization point and are done by asynchronous processing using output threads in the control region. The output thread runs as a service request block (SRB): a separate dispatchable MVS task. You can specify up to 255 output threads. This means that:
  - The CICS task can be freed earlier
  - Parallel processing is increased and throughput on multiprocessors is improved.
- Less I/O

  The cost of I/O per SDEP segment inserted can be very low because SDEP segments are gathered in one buffer and are written out only when it is full. This means that many transactions can "share the cost" of SDEP CI writes to a DEDB. SDEPs should have larger CIs to reduce I/Os.
- Reduced logging overhead

  DEDB log buffers are written to OLDS only when they are full. This means less I/O than would be needed with full function databases.

High speed sequential processing (HSSP)

Using DBCTL enables you to use high speed sequential processing (HSSP). HSSP is useful with applications that do large scale sequential updates to DEDBs, which may require an image copy after the DEDBs are updated. Using HSSP provides the following major benefits:

- DEDB processing time can be improved by using the IBM 3990 Storage Control Model 3 Fast Write capability and the IBM 3990 Storage Control Model 3 Sequential Mode for both READs and WRITEs.
- You can take an HSSP image copy during a sequential update job. This avoids having to make a subsequent sequential pass through the DEDB areas to take an image copy.
- HSSP reduces elapsed DEDB processing time by using private buffer pools and optimizing locking.
- Only a minimum amount of log data is written to the IMS system log when you request an HSSP image copy. This reduces the large amount of logging that such large scale sequential runs usually involve.

For further guidance on HSSP, see *IMS Release Planning Guide*.

IMS asynchronous database buffer purge facility

IMS includes the asynchronous database buffer purge facility.

At syncpoint time, when database buffers are to be flushed, buffers that are to be written to different devices are written concurrently, rather than serially, as in earlier releases of IMS. For further guidance, see *IMS System Administration Guide*.

The asynchronous database buffer purge facility should improve response time for transactions that update databases on multiple devices in a single UOW.

Virtual storage usage

CICS regions that previously used local DL/I can obtain considerable virtual storage constraint relief because the following storage areas reside in the DBCTL address spaces: all DL/I and DBRC code and control blocks, OSAM and VSAM buffer pools and related control blocks, PSB, DMB, and ENQ pools.

However, DBCTL requires some MVS CSA storage, which can lower the maximum available region size in the MVS system. See *IMS System Administration Guide* for details of CSA and other DBCTL storage requirements.

Improved throughput on multiprocessors

You can obtain throughput improvements on multiprocessors when using IMS Version 12 or later by using the CICS open transaction environment (OTE), providing that the application code is threadsafe.

You can obtain further performance improvements by using DEDBs instead of full-function databases. See "Access to data entry databases (DEDBs)" on page 8 for introductory guidance on DEDBs, and "Using DEDBs" on page 157 for information about the performance aspects.

Notices

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

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

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

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

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

The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

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

Licensees of this program who 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 United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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

Programming Interface Information

This book is intended to help you evaluate, install, and use the CICS-IMS Database Control (DBCTL) interface.

This book also documents Product-sensitive Programming Interface and Associated Guidance Information and Diagnosis, Modification or Tuning Information provided by CICS.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of CICS. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs, by an introductory statement to a chapter or section.

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

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

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

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

Trademarks

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.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Bibliography

CICS books for CICS Transaction Server for z/OS

General

CICS Transaction Server for z/OS Program Directory, GI13-0565
CICS Transaction Server for z/OS What's New, GC34-7192
CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.1, GC34-7188
CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.2, GC34-7189
CICS Transaction Server for z/OS Upgrading from CICS TS Version 4.1, GC34-7190
CICS Transaction Server for z/OS Installation Guide, GC34-7171

Access to CICS

CICS Internet Guide, SC34-7173
CICS Web Services Guide, SC34-7191

Administration

CICS System Definition Guide, SC34-7185
CICS Customization Guide, SC34-7161
CICS Resource Definition Guide, SC34-7181
CICS Operations and Utilities Guide, SC34-7213
CICS RACF Security Guide, SC34-7179
CICS Supplied Transactions, SC34-7184

Programming

CICS Application Programming Guide, SC34-7158
CICS Application Programming Reference, SC34-7159
CICS System Programming Reference, SC34-7186
CICS Front End Programming Interface User's Guide, SC34-7169
CICS C++ OO Class Libraries, SC34-7162
CICS Distributed Transaction Programming Guide, SC34-7167
CICS Business Transaction Services, SC34-7160
Java Applications in CICS, SC34-7174

Diagnosis

CICS Problem Determination Guide, GC34-7178
CICS Performance Guide, SC34-7177
CICS Messages and Codes Vol 1, GC34-7175
CICS Messages and Codes Vol 2, GC34-7176
CICS Diagnosis Reference, GC34-7166
CICS Recovery and Restart Guide, SC34-7180
CICS Data Areas, GC34-7163
CICS Trace Entries, SC34-7187
CICS Debugging Tools Interfaces Reference, GC34-7165

Communication

CICS Intercommunication Guide, SC34-7172
CICS External Interfaces Guide, SC34-7168

Databases

CICS DB2 Guide, SC34-7164
CICS IMS Database Control Guide, SC34-7170

CICS Shared Data Tables Guide, SC34-7182

CICSPlex SM books for CICS Transaction Server for z/OS

General

CICSPlex SM Concepts and Planning, SC34-7196
CICSPlex SM Web User Interface Guide, SC34-7214

Administration and Management

CICSPlex SM Administration, SC34-7193
CICSPlex SM Operations Views Reference, SC34-7202
CICSPlex SM Monitor Views Reference, SC34-7200
CICSPlex SM Managing Workloads, SC34-7199
CICSPlex SM Managing Resource Usage, SC34-7198
CICSPlex SM Managing Business Applications, SC34-7197

Programming

CICSPlex SM Application Programming Guide, SC34-7194
CICSPlex SM Application Programming Reference, SC34-7195

Diagnosis

CICSPlex SM Resource Tables Reference Vol 1, SC34-7204
CICSPlex SM Resource Tables Reference Vol 2, SC34-7205
CICSPlex SM Messages and Codes, GC34-7201
CICSPlex SM Problem Determination, GC34-7203

Other CICS publications

The following publications contain further information about CICS, but are not provided as part of CICS Transaction Server for z/OS, Version 4 Release 2.

Designing and Programming CICS Applications, SR23-9692

CICS Application Migration Aid Guide, SC33-0768

CICS Family: API Structure, SC33-1007

CICS Family: Client/Server Programming, SC33-1435

CICS Family: Interproduct Communication, SC34-6853

CICS Family: Communicating from CICS on System/390, SC34-6854

CICS Transaction Gateway for z/OS Administration, SC34-5528

CICS Family: General Information, GC33-0155

CICS 4.1 Sample Applications Guide, SC33-1173

CICS/ESA 3.3 XRF Guide , SC33-0661

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully.

You can perform most tasks required to set up, run, and maintain your CICS system in one of these ways:

- using a 3270 emulator logged on to CICS
- using a 3270 emulator logged on to TSO
- using a 3270 emulator as an MVS system console

IBM Personal Communications provides 3270 emulation with accessibility features for people with disabilities. You can use this product to provide the accessibility features you need in your CICS system.

Index

Special characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Readers' Comments — We'd Like to Hear from You

CICS Transaction Server for z/OS
Version 4 Release 2
IMS Database Control Guide

Publication No. SC34-7170-01

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: +44 1962 816151
- Send your comments via email to: idrcf@uk.ibm.com

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	Email address

Readers' Comments — We'd Like to Hear from You
SC34-7170-01

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM United Kingdom Limited
User Technologies Department (MP095)
Hursley Park
Winchester
Hampshire
United Kingdom
SO21 2JN

Fold and Tape | **Please do not staple** | Fold and Tape

SC34-7170-01

Cut or Fold
Along Line

SC34-7170-01