CICS Transaction Server for z/OS

CICS Diagnosis Reference

Version 3 Release 1

GC34-6899-03

CICS Transaction Server for z/OS

CICS Diagnosis Reference

Version 3 Release 1

GC34-6899-03

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 1571.

This edition applies to Version 3 Release 1 of CICS Transaction Server for z/OS, program number 5655-M15, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Contents

Preface

What this book is about

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

”MVS” is used for the operating system, which is an element of z/OS.

This book gives a detailed description of the various components that make up a CICS system. It also provides reference tables of CICS source modules and executable modules.

This book is intended to help you in diagnosing problems with CICS.

Who this book is for

This book provides a basis for communication between the system programmer and the IBM® support representative whenever a problem with CICS code is suspected.

What you need to know to use this book

You should have system programming experience and a good working knowledge of CICS and of the functions used in your system to support CICS applications.

Before using this book, you should have read the *CICS Problem Determination Guide* to learn about the general approach to CICS problem-solving and the procedures to use when diagnosing and reporting system problems. You should already be familiar with the general layout of CICS traces and dumps.

In addition, you may need to refer to the following books in the CICS library while diagnosing what appears to be a system problem:

- The *CICS Data Areas* manual for details of the layout and contents of CICS data areas
- The *CICS Messages and Codes* manual for information about the messages and abend codes that can be issued by a running CICS system

Notes on terminology

The following abbreviations are used throughout this book:

Term	Meaning
CICS	When used without qualification in the book, refers to the CICS element of IBM CICS Transaction Server for z/OS
ESA	IBM Enterprise Systems Architecture/370 (ESA/370)
MVS™	The IBM operating system, which can be either an element of OS/390®, or MVS/Enterprise System Architecture System Product (MVS/ESA SP)
VTAM®	IBM Advanced Communications Function/Virtual Telecommunications Access Method (ACF/VTAM)
VTAM/NCP	IBM Virtual Telecommunications Access Method/Network Control Program (VTAM/NCP)
IMS™	IMS
DL/I	The DL/I facilities of IMS
FEPI	Front End Programming Interface

Summary of changes

This book is based on the CICS Diagnosis Reference for CICS Transaction Server for z/OS, Version 2 Release 1, LY33-6088-31. Changes from that edition are marked by vertical bars in the left margin.

This part lists the changes that have been made for the following recent releases:

Changes for CICS Transaction Server for z/OS Version 3 Release 1

The more significant changes for this edition are:

Technical changes

-

Structural changes

- Part 3, "CICS domains," on page 511 is new, and contains information about CICS domain interfaces. This information was previously in Part 2, "CICS components," on page 9.

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

The more significant changes for this edition are:

Technical changes

- The following have been added:
  Chapter 93, "Object Transaction Service domain (OT)," on page 939
  Chapter 18, "ECI over TCP/IP," on page 131
  Chapter 84, "IP ECI (IE) domain," on page 811
- There is new and changed information in:
  Chapter 70, "Application domain (AP)," on page 513
  Chapter 113, "Transaction manager domain (XM)," on page 1275
  Chapter 8, "CICS-DB2 Attachment Facility," on page 79
  Chapter 80, "Dispatcher domain (DS)," on page 697
  Chapter 82, "Enterprise Java domain (EJ)," on page 753
  Chapter 85, "IIOP domain (II)," on page 815
  Chapter 103, "JVM domain (SJ)," on page 1127
  Chapter 88, "Log manager domain (LG)," on page 867
  Chapter 95, "Program manager domain (PG)," on page 949
  Chapter 114, "Security manager domain," on page 1313
  Chapter 44, "Statistics utility program (DFHSTUP)," on page 353
  Chapter 111, "User domain," on page 1241

Structural changes

- Information about trace entries have been moved to *CICS Trace Entries*.
- Chapter 31, "Language Environment interface," on page 309 has moved to its present position, to reflect its new title; the term *Language Environment* is now used in place of *CICS-AD/Cycle Language Environment/370*.

Changes for CICS Transaction Server for z/OS, Version 2 Release 1

The main changes are as follows:

- New chapters on the new domain components of CICS were added:
  - Log manager domain, starting on page 867.
  - VTAM generic resources, starting on page 473.
  - Recovery manager domain, starting on page 1061.

- The file control chapter has been amended to show new function within file control.
- The following chapters have been deleted:
  - Asynchronous processing.
  - Dynamic backout programming.
  - Emergency restart.
  - Journaling chapters and system log/journaling utilities - this is largely replaced by the Log manager domain.
  - Local DL/I.
  - Shared databases.
  - Time of day control.
  - Volume control.

Part 1. Introduction

This book describes the functional areas (or components) into which CICS is divided. To understand more about a particular functional area, use the contents list and the index to find the appropriate information.

If you are using this book to diagnose a system problem, to find out whether a function is **working as designed**, you should also consult the special topic, administration, or programming books in the CICS Transaction Server for z/OS library.

In this and other CICS books, the word “component” is used in a general way to refer to any unit of code that performs an identifiable set of functions and manages a certain type of data.

Some CICS components are shipped as **object code only (OCO)**. If the component causing a problem is OCO, it is the responsibility of IBM to diagnose the problem further. If the component is not OCO, you can refer to the source code on microfiche, and use the detailed information in this book to identify more specifically the cause of the problem. The Chapter 115, “CICS directory,” on page 1343 shows which CICS object modules are regarded as OCO; no source code is available for these modules.

Chapter 1. CICS domains

At the top level, CICS is organized into **domains**. With the exception of the application domain, which contains several components, each domain is a single major component of CICS. Domains never communicate directly with each other. Calls between domains are routed through kernel linkage routines. Calls can be made only to official interfaces to the domains, and they must use the correct protocols. This structure is shown in Figure 1.

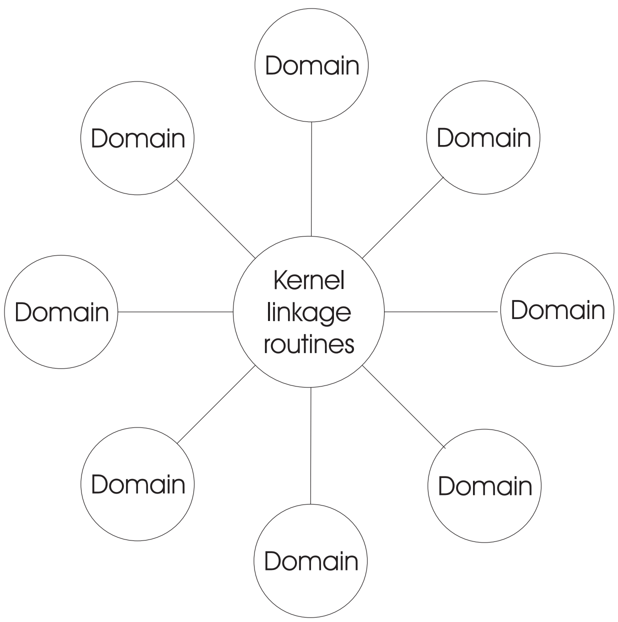


Figure 1. CICS organization—domains

Each domain manages its own data. No domain accesses another domain's data directly. If a domain needs data belonging to another domain, it must call that domain, and that domain then passes the data back in the caller's parameter area.

The following table lists the CICS domains alphabetically by domain identifier. For each domain, the table also shows whether or not the domain is OCO, and gives a page reference to the section describing the interfaces to the domain.

Domain ID	Domain	OCO?	See page
AP	Application	See note	513
BA	Business Application Manager	Yes	603
CC	Local catalog	Yes	633
DD	Directory manager	Yes	641
DH	Document handler	Yes	647
DM	Domain manager	Yes	663
DP	Debugging profile domain	Yes	673
DS	Dispatcher	Yes	697
DU	Dump	No	721
EJ	Enterprise Java™	No	753
EM	Event manager	Yes	799
GC	Global catalog	Yes	633
IE	IP ECI	Yes	811
II	IIOP	No	815
KE	Kernel	Yes	831

Domain ID	Domain	OCO?	See page
LD	Loader	Yes	853
LG	Log manager	Yes	867
LM	Lock manager	Yes	897
ME	Message	Yes	901
MN	Monitoring	Yes	911
NQ	Enqueue	Yes	921
OT	Object transaction service	No	939
PA	Parameter manager	Yes	945
PT	Partner	Yes	1051
PG	Program manager	Yes	949
RM	Recovery manager	Yes	1061
RX	Resource recovery service	Yes	1101
RZ	Request Stream	No	1105
SH	Scheduler services	Yes	1119
SJ	JVM Domain	No	1275
SM	Storage manager	Yes	1143
SO	Sockets Domain	No	1175
ST	Statistics	Yes	1195
TI	Timer	Yes	1201
TR	Trace	No	1205
TS	Temporary storage	Yes	1221
US	User	Yes	1241
WB	Web	No	1257
XM	Transaction manager	Yes	1275
XS	Security manager	Yes	1313

Note: The application domain is mainly non-OCO, but it contains these OCO components:
- CICS data table services
- RDO for VSAM files and LSR pools
- Some EXEC CICS system programming functions
- Autoinstall terminal model manager
- Partner resource manager
- SAA Communications and Resource Recovery
- Some of the file control functions
- Recovery manager connectors interfaces.

The offline statistics utility program (DFHSTUP) and the system dump formatting routines are also treated as OCO.

Domain gates

A **domain gate** is an entry point or interface to a domain. It can be called by any authorized caller who needs to use some function provided by the domain.

A number of domain functions are available through the exit programming interface (XPI). For details, see the CICS Customization Guide.

In practice, every domain has several gates. Each gate has a 4-character identifier; the first two characters are the identifier of the owning domain, and the second two characters differentiate between the functions of the domain's gates. Here, for example, are two of the dispatcher (DS) domain's gates:

DSAT
DSSR

Functions provided by gates

An individual gate can provide many functions. The required function is determined by the parameters included on the call. The DSSR gate of the DS domain, for example, provides all these functions:

ADD_SUSPEND
DELETE_SUSPEND
INQUIRE_SUSPEND_TOKEN
RESUME
SUSPEND
WAIT_MVS
WAIT_OLDC
WAIT_OLDW.

Specific gates, generic and call-back gates

It is useful to distinguish between **specific gates**, **generic gates** and **callback gates**:

- A specific gate gives access to a set of functions that are provided by that domain alone. The functions are likely to be requested by many different callers.

  DS domain, for example, has a specific gate (DSAT) that provides CHANGE_MODE and CHANGE_PRIORITY functions (among other functions). Only the DS domain provides those functions, but they can be requested by many different callers.
- A generic gate gives access to a set of functions that are provided by several domains.

  Most domains provide a QUIESCE_DOMAIN function, for example, so that they can be quiesced when CICS is shutting down normally. They each have a generic gate that provides this function. DM domain makes a **generic call** to that gate in any domain that is to be quiesced.
- A call-back gate also gives access to a set of functions that can be provided by several domains. Unlike a generic gate where the call is broadcast to all domains that have provided a gate a call-back is restricted to specific domains but uses a format owned by the calling domain.

  For example the Recovery Manager calls the domains that have registered an interest in syncpoint processing using the PERFORM_PREPARE function format that it owns.

Domain call formats

Any module calling a domain gate must use the correct **format** for the call. The format represents the parameter list structure. It describes the parameters that must be provided on the call (the **input** parameters), and the parameters that are returned to the caller when the request has been processed (the **output** parameters).

For example, Table 1 lists the input and output parameters for the ATTACH function of the DS domain's DSAT gate.

Table 1. Domain call formats

Input parameters	Output parameters
PRIORITY USER_TOKEN [TIMEOUT] TYPE [MODE] [TASK_REPLY_GATE_INDEX] [SPECIAL_TYPE]	TASK_TOKEN RESPONSE [REASON]

Parameters not shown in brackets are mandatory, and are always interpreted in trace entries. Parameters shown in brackets are optional, and are in trace entries only if values have been set. An exception to this rule is that, regardless of whether REASON is mandatory or optional for a particular function, its value is included in a trace entry only for a non-'OK' response.

The domain call formats described are in the sections dealing with the domains that own them, as discussed in “Ownership of formats.”

Ownership of formats

Every format is ‘owned’ by a domain:

- The formats for specific calls are owned by the domain being called. DS domain, for example, owns the format for the CHANGE_MODE and CHANGE_PRIORITY calls. This book uses the term **specific format** to refer to such formats.
- The formats for generic calls and call-back calls are owned by the calling domain. DM domain, for example, owns the format for calls to (generic) gates providing the QUIESCE_DOMAIN function in other domains. This book uses the term **generic format** to refer to such formats.

Tokens

Tokens are passed as parameters on many domain calls. They identify uniquely objects that are operands of domain functions.

Here are some examples:

TASK_TOKEN
uniquely identifies a task to be used as the operand of a function.

DOMAIN_TOKEN
uniquely identifies a domain to be used as the operand of a function.

SUSPEND_TOKEN
uniquely identifies a task for the purpose of a suspend or resume dialog.

Responses

On all domain calls, one of the output parameters is the domain’s response to the call. It can have any one of these values:

OK
When a domain call succeeds, a response of ‘OK’ is given and the REASON code is not set. The requested function has been completed successfully.

EXCEPTION
Processing of the function could not be completed for the reason specified in the REASON field. The domain state remains unchanged if such an error occurs.

DISASTER
The domain could not complete the request because of some irrecoverable system problem. If there is a major error in the domain, this is reported.

INVALID
The parameter list is not valid. If a call is used incorrectly, this is reported.

KERNERROR
The kernel was unable to call the required function gate.

PURGED
A purge has been requested for the task making the domain call.

Chapter 2. Application domain

Application programs are run in the application (AP) domain, which contains several major components, as shown in Figure 2.

Most application domain CICS functions are either provided by modules that are part of the CICS nucleus, that is to say they are an integral part of the system and are loaded at system initialization time, or they are system application programs, which are loaded as needed in the same way as user application programs.

- AP domain
  - Application services
    - Basic mapping support
    - Built-in functions
    - Command interpreter
    - Data interchange program
    - Execution diagnostic facility
  - System reliability
    - Abnormal condition program
    - Dynamic backout
    - Emergency restart
    - Keypoint programs
    - Node abnormal condition program
    - Node error program
    - Program error program
    - Retry program
    - System recovery program
    - Task-related user exit recovery
    - Terminal abnormal condition program
    - Terminal error program
  - Extended recovery facility
  - Intercommunication facilities
    - Distributed transaction processing
    - Function shipping
    - Interregion communication
    - Transaction routing
    - Recovery manager connectors
    - VTAM LU6.1
    - VTAM LU6.2
  - System control
    - AP domain initialization
    - AP domain termination
    - DL/I and DBCTL support
    - EXEC interface program
    - File control
    - Interval control
    - Resource recovery manager
    - Storage compatibility
    - Syncpoint program
    - Table manager
    - Task-related user exit control
    - Temporary-storage control
    - Terminal control
    - Trace compatibility
    - Transient data control
    - User exit control
  - System services
    - Dynamic allocation
    - Field engineering program
    - "Good morning" message program
    - Master terminal program
    - Massage switching
    - Operator terminal
    - Resource definition
    - Signon and sign-off
    - System spooling interface
    - Time-of-day control

Figure 2. AP domain—major components

Part 2. CICS components

This part describes the major components of a CICS system that do not use a domain interface. Offline utilities, such as the statistics utility program, are also covered.

Sections are ordered alphabetically by component name for quick reference.

Chapter 3. Autoinstall for terminals, consoles and APPC connections

Autoinstall for terminals provides the ability to log on to CICS from a logical unit (LU), known to VTAM but not previously defined to CICS, and to make a connection to a running CICS system.

A new connection is created and installed automatically if autoinstall for connections is enabled, and either of the following occurs:

- An APPC BIND request or CINIT request is received for an APPC service manager (SNASVCMG) session that does not have a matching CICS CONNECTION definition
- A BIND is received for a single session that does not have a matching CICS CONNECTION definition.

A new console is created and installed automatically if autoinstall for consoles is enabled and a CIB (Command Input Buffer sent from MVS) is received by CICS (DFHZCNA) and the console TCTTE does not already exist.

For an introduction to autoinstall, and information about how to implement it, see the *CICS Resource Definition Guide*.

The *CICS Customization Guide* gives information about implementing the autoinstall user program. The CICS-supplied programs are:

- DFHZATDX, which provides autoinstall for terminals only
- DFHZATDY, which provides autoinstall for terminals and APPC connections.

These programs are user-replaceable, because you may need to tailor the basic function to suit your CICS environment.

Design overview

Before a VTAM device can communicate with CICS, a VTAM session must be established between the device and CICS. The sequence of operations is LOGON, Open Destination (OPNDST), and Start Data Traffic (SDT). CICS can also initiate the LOGON by using a SIMLOGON.

The session can be requested by:

- Specifying AUTOCONNECT when the terminal is defined to CICS
- A VTAM master terminal command requesting a LOGON to CICS for a given terminal; for example, V NET,LOGON=CICSA,ID=L3277C1
- An individual terminal operator issuing a LOGON request (LOGON APPLID(CICSA))
- A CICS master terminal command requesting LOGON for a given terminal (CEMT SET TERMINAL(xxxx) INSERVICE ACQUIRED)
- CICS internally requesting a LOGON; for example, to process an ATI request
- LOGAPPL=CICS in the LU statement.

Consoles are not VTAM resource but they usse a similar mechanism to autoinstall the TCTTE.

Autoinstall of a terminal logon flow

This section describes the flow of control for a terminal that is to be logged on by autoinstall.

1. When a terminal or single session APPC device attempts to log on, VTAM drives the **logon exit**. The CICS logon exit is DFHZLGX (load module DFHZCY).

   In the following circumstances, an LU is a candidate for autoinstall:

   - If it is not already defined to CICS (using RDO)

- If neither CICS nor VTAM is quiescing
- If the autoinstall user program (specified by the AIEXIT system initialization parameter) exists
- If the VTAM RPL is present
- If it is not an LU6.1 session or an LU6.2 parallel session
- If it is an LU6.2 single session terminal and the ISC=YES system initialization parameter is specified
- If the maximum number of concurrent logon requests (specified by the AIQMAX system initialization parameter) has not been exceeded.

DFHZLGX searches for the terminal in the terminal control table (TCT) by comparing the NETNAME passed by VTAM with the NETNAME found in the NIB descriptor for each installed terminal.

If a match is not found and AUTOINSTALL is enabled (TCTVADEN is set), CICS verifies that the terminal is eligible for autoinstall. Processing then consists of:

- Building an autoinstall work element (AWE) by issuing an MVS GETMAIN for subpool 1
- Copying the CINIT RU into the AWE
- Adding the AWE to the end of the AWE chain, which is chained from the TCT prefix.

If a match is found showing that this autoinstall terminal already exists, a postponed work element (PWE) is created and the terminal is reinstalled after deletion of the TCTTE (TCTEDZIP is ON) or if AILDELAY=0. If, however, AILDELAY¬=0 but TCTEDZIP is not ON (that is, the TCTTE deletion is pending), the TCTTE is reused after cleanup.

2. Later, the work element (AWE) is actioned by DFHZACT attaching transaction CATA. For every AWE on the AWE chain, the DFHZATA autoinstall program is dispatched, passing to DFHZATA the AWE’s address.
3. The DFHZATA program:
   a. Validates the BIND image in the CINIT RU. If the image is not valid, issue message DFHZC6901.
   b. If VTAM Model Terminal Support (MTS) is being used (ACF/VTAM 3.3 or later), and the name of a CICS model has been supplied in a X'2F' MTS control vector, DFHZATA checks that the model exists by using the AIIQ subroutine interface of the AITM manager (see Chapter 4, “Autoinstall terminal model manager,” on page 23). If the model does not exist, issue message DFHZC6936.

      DFHZATA compares the BIND image contained in the MTS model with the BIND image passed in the CINIT RU. If there is a mismatch, issue message DFHZC6937.

      This validated MTS model is the only model passed to the autoinstall control program.
   c. In the absence of an MTS model name, DFHZATA browses the autoinstall terminal model (AITM) table using the AIIQ subroutine interface of the AITM manager. These models must have been installed, with appropriate TYPETERM definitions, either at system initialization or by a CEDA INSTALL command.

      Compare the BIND image contained in each model with the BIND image passed in the CINIT RU, and build a list of suitable models to be passed to the autoinstall control program.

      For autoinstall of an LU to be successful, the following ***must match***:
      - CINIT BIND image, taken from the VTAM LOGMODE entry specified for the LU in the VTAMLST
      - Autoinstall terminal model BIND image, built according to the specifications in the TYPETERM and TERMINAL definitions.

      (Both versions of the BIND image should accurately define the characteristics of the device.) If the model BIND matches the CINIT BIND, the model is added to the list of candidate entries.

      If the list is empty (no matching models are found), the request is rejected and message DFHZC6987 is written to the CADL log.
   d. On completion of the model search, if any, DFHZATA links to the autoinstall control program (the CICS-supplied default is DFHZATDX).
   e. Issue DFHZCP_INSTALL to create the TCTTE. DFHZATA uses information from the model selected by the exit program and the associated TYPETERM entry to build the TCTTE.

f. If the install was successful, commit the TCTTE and queue it for LOGON processing. The new TCTTE is queued for OPNDST processing, then later the “good morning” message is written.

g. Free the AWE.

Autoinstall of APPC device logon flow

This section describes the flow of control for an APPC parallel session device (or single session via a BIND) that is to be logged on by autoinstall.

1. When an APPC device attempts to logon, VTAM drives the logon exit DFHZLGX if a CINIT is received, or the SCIP exit DFHZBLX if a BIND is received.

   Note that DFHZBLX is a new VTAM exit module that is called by DFHZSCX if an LU62 BIND has been received.

   In the following circumstances, an APPC LU is a candidate for autoinstall.
   - If the connection is not already defined to CICS.
   - If the connection is not already installed.
   - If the autoinstall user program (specified by the AIEXIT system initialization parameter) exists and caters for functions 2-4 as well as functions 0-1.
   - If the VTAM ACB is open.
   - If it is an APPC parallel session connection.
   - If it is an APPC single session connection with an incoming BIND (as opposed to CINIT - which uses terminal autoinstall).
   - If ISC=YES is specified in the SIT.
   - If the maximum number of concurrent logon requests (specified by AIQMAX) has not been exceeded.
   - If the customer has installed the correct 'template' connection that is to be 'cloned' (or copied) to create the new connection.

   DFHZLGX or DFHZBLX searches for the connection in the terminal control table (TCT) by comparing the NETNAME passed by VTAM with the NETNAME found in the NIB descriptor for each installed session.

   If a match is found and AUTOINSTALL is enabled (TCTVADEN is set), CICS verifies that the terminal is eligible for autoinstall. Processing then consists of:
   - Building an autoinstall work element (AWE) by issuing an MVS GETMAIN for subpool 1.
   - Copying the CINIT RU (DFHZLGX) or BIND (DFHZBLX) into the AWE.
   - Adding the AWE to the end of the AWE chain, which is chained from the TCT prefix.

   If a match is found showing that this connection already exists then the logon proceeds as for a defined connection.
2. Later, the AWE is actioned by DFHZACT attaching transaction CATA. For every AWE on the AWE chain, the DFHZATA autoinstall program is dispatched, passing to DFHZATA the AWE's address.
3. The DFHZATA program:
   a. Validates the BIND image passed in the AWE. If the image is not valid, issue message DFHZC6901.

   b. Calls DFHZGAI Function(CREATE_CLONE_BPS) to create a Builder Parameter Set from which to create the new connection ('clone'). This is done by calling the customer supplied autoinstall user exit program (which can be based on DFHZATDY) in which the customer chooses which 'template' connection the new connection should be copied from.

      If at any point DFHZGAI finds a problem it issues message DFHZC6920 or DFHZC6921 or DFHZC6922 with an exception trace entry which will explain the reason for failure.

   c. Issue DFHZCP function(INSTALL) to create the CONNECTION, MODEGROUP and SESSIONs, based on the attributes of the template connection.

d. For parallel sessions with an incoming BIND, chose the SNASVCMG secondary session and call DFHZGAI (SET_TCTTE_FOR_OPNDST). This mimics code in DFHZBLX to check the session against the incoming BIND.

   If at any point DFHZGAI finds a problem it issues message DFHZC6923 with an exception trace entry which explains the reason for failure.

e. For parallel session with an incoming CINIT, chose the SNASVCMG primary session.

f. If the install was successful, commit the CONNECTION and queue it for logon processing. The new CONNECTION is queued for OPNDST processing.

g. Free the AWE.

Autoinstall of an APPC Generic Resource connection

If this system is registered as a generic resource and a bind is received from another generic resource then VTAM exit DFHZBLX will initiate an autoinstall if there is no generic or member name connection available for use.

An AWE is created with extra parameters such as the generic resource name and member name of the partner and possibly a suggested template.

Autoinstall then continues as for normal APPC and the extra parameters are reflected into the TCSE and TCTTE via the BPS.

Autoinstall of consoles install flow

1. The modify command comes into DFHZCNA via a CIB (Command Input Buffer) from MVS when a user types a console command for CICS.
2. DFHZCNA scans the Console Control Elements for a matching console name. If no CCE is found and autoinstall for consoles is enabled then an Autoinstall Work Element is created and added to the AWE queue.
3. DFHZACT scans the AWE queue and attached the CATA transaction.
4. The CATA transaction calls DFHZATA which sees the AWE is fir a console (sometimes called a Console Work Element) and calls DFHZATA2.
5. DFHZATA2 does the following:
   a. Finds the console models (AICONS is supplied in group DFHTERMC).
   b. If SIT AICONS(YES) is specified the models are passed to the autoinstall user-replaceable program which returns the termid. The default autoinstall user-replaceable program returns the last 4-characters of the consolename.
   c. If SIT AICONS(AUTO) is specified DFHZGBM is called to get a name in the console bitmap in the form ^AAA. The autoinstall user-replaceable program is not called.
   d. Calls DFHZCP FUNCTION(INSTALL).
   e. Issues EXEC CICS SYNCPOINT.
   f. Signs on if using preset security of USERID=*EVERY|*FIRST specified in the AI model TYPETERM.
   g. Geta a TIOA to hold the data specified in the command, e.g. if /f jobname,CEMT I TE was typed at the console then CEMT I TE is put into the TIOA.
   h. Call DFHZATT to attach the transaction specified in the MODIFY command (e.g. CEMT).

Sign-on to consoles flow

If a CIB is received with the same console name but with a different USERID then the autoinstall program DFHZATA2 is called to sign off the original USERID and sign on to the new USERID as follows:

1. DFHZCNA receives the modify and
   a. Finds the CCE

   b. Finds that the USERID is different and is already signed on
   c. Creates an AWE for signoff/on
   d. Chains the AWE for DFHZACT.
2. DFHZACT attaches CATA
3. CATA calls DFHZATA which calls DFHZATA2 for signoff/on
4. DFHZATA2 issues preset security sign off for the original USERID followed by sign on for the new USERID
5. DFHZATA2 then gets a TIOA for the modify command data and calls DFHZATT to attach the transaction as for normal autoinstall for consoles.

Disconnection flow for terminals (LU-initiated)

This section describes the flow of control when a request is made to disconnect an autoinstalled terminal (for example, by entering a CESF LOGOFF command), ultimately causing an EXEC CICS ISSUE LOGOFF command to be issued.

1. First the following functions are performed:
   - Set on the CLSDST flag in the TCTTE.
   - Put the TCTTE on the **activate chain** for DFHZACT to dispatch.
2. Control is then passed to the **Close destination program**, DFHZCLS, which performs the following functions:
   - Set on the SHUTDOWN_IN_PROGRESS flag in the TCTTE.
   - Set on the REQUEST_SHUTDOWN flag in the TCTTE.
3. The **Send asynchronous commands program**, DFHZDSA is then called to send a VTAM SHUTD command to the LU (autoinstalled terminal) to be disconnected. The DFHZDSA program removes the TCTTE from the activate chain, pending completion of the SHUTD command.
4. When the VTAM SHUTD command has completed, VTAM calls the **asynchronous send exit**, DFHZSAX, which performs the following functions:
   - Set off the REQUEST_SHUTDOWN flag in the TCTTE.
   - Set on the SHUTDOWN_SEND flag in the TCTTE.
   - Put the TCTTE back on the activate chain for DFHZACT to dispatch.
5. VTAM then drives the **asynchronous receive exit**, DFHZASX, with the SHUTC (“shutdown complete”) command sent by the LU to be disconnected. DFHZASX performs the following functions:
   - Ensures that the NODE_QUIESCED_BY_CICS, SHUTDOWN_IN_PROGRESS, and CLSDST flags are still on.
   - Puts the TCTTE back on the activate chain for DFHZACT to dispatch.
6. Control is then passed to the **Close_Destination program**, DFHZCLS. The DFHZCLS program performs the following functions:
   - Set on the PENDING_DELETE flag in the TCTTE to prevent VTAM exits scheduling requests for the device.
   - Issue UNBIND (CLSDST POST=RESP) for the device.
7. The **Close destination exit**, DFHZCLX, is driven. If the CLSDST request is successful (that is, there is a positive response from UNBIND), the following functions are performed:
   - Set on the SESSION_CLOSED flag in the TCTTE.
   - Flag the TCTTE for deletion.
   - Enqueue the TCTTE to DFHZNAC.
8. Control is passed to the DFHZNAC program, which performs the following functions:
   - Set on the DELETE_REQUIRED flag in the TCTTE.
   - Put the TCTTE on the activate chain for DFHZACT to dispatch.
   - Issue message DFHZC3462 (session terminated).

9. On the delete request, the DFHZNCA copybook of DFHZNAC checks the value of the system initialization parameter AILDELAY.
   - If AILDELAY is zero, the TCTTE is queued via DFHZACT with the address of the TCTTE as input. Its function is to perform cleanup operations, the principal operation being to ask DFHZCQ to delete the TCTTE.
   - If AILDELAY is not zero, DFHZNCA initiates CATD using the delay specified and passes the address of the TCTTE.

   Up to three attempts are made to delete the TCTTE. This is because the reason for the failure may be the existence of a transient condition, such as the TCTTE being on the DFHZNAC queue to output a message to CSMT. If the initial delete attempt fails, it is attempted again after one second; if this fails, another attempt is made after a further 5 seconds. If the third attempt fails, it is assumed that the failure is a hard failure, which will not disappear until the device is reconnected; in this case, message DFHZC6943 is issued, a syncpoint is taken, and the TCTTE delete status is reset to make the TCTTE reusable.

   If the deletion is successful, the delete is committed, the autoinstall control program is invoked to permit any specific cleanup operations to take place, and message DFHZC6966 is issued.

   If a PWE exists for this TCTTE, the PWE is requeued onto the AWE chain.

Disconnection of an autoinstalled terminal can also be requested by CICS shutdown, terminal time-out, and terminal errors. In these cases the flow is slightly different.

Deletion of autoinstalled APPC devices.

This section describes the flow of control when an APPC sync level 1 device has its last session released. This can occur as a result of unbind flows from the partner or a RELEASE command being issued against the connection in this system.

Only synclevel 1 autoinstalled connections are deleted in this way. They will have had TCSE_IMPLICIT_DELETE set by the builders from zx_delete_x in the BPS (set by DFHZGAI).

TCSE_CATLG_NO indicates that the connection is not to be written to the catalogue (SIT Parameter AIRDELAY=0).

1. After DFHZCLS, the CLSDST program, issues DFHTCPLR TIDYUP TCSEDDP and TCSE_DELETE_SCHEDULE are set and CATD is initiated with a delay of AILDELAY.
2. CATD runs DFHZATD which sets TCSE_DELETE_STARTED and calls DFHZCP FUNCTION=DELETE to delete the sessions, modegroup and connection.

If a SIMLOGON or BIND occur before the delete actually starts (TCSE_DELETE_SCHEDULED) then the connection delete is aborted and the connection reused.

If a SIMLOGON occurs during the actual delete (TCSE_DELETE_STARTED) then the delete is vetoed and the connection is reacquired.

If a BIND occurs during the actual delete (TCSE_DELETE_STARTED) then the delete goes ahead and the PWE that was created is turned into an AWE and the logon will create a new connection.

If TCSE_DELETE_AT_RESTART is set then DFHZATR will delete the connection if it has not been used after restart with a delay specified in the SIT AIRDELAY parameter.

Disconnection flow (APPC devices)

These connections are not deleted at LOGOFF time, so the disconnection flow is the same as for a defined connection.

Deletion of autoinstalled consoles

Consoles are deleted after a certain period of inactivity. The default is 60 minutes but this can be overridden in the autoinstall user-replaceable program.

1. The delete time is saved in the CCE during install in TCTCE_TIMEOUT_TIME.
2. DFHCESC runs at certain intervals
3. DFHCESC checks the CCEs for any console whose delete time has expired
4. For each expired CCE DFHCESC does the following
   a. Attaches CATD to do the delete
   b. CATD calls DFHZATD as for a terminal

Shipping a TCTTE for transaction routing

For transaction routing, a terminal can be defined by an entry in the terminal-owning region (TOR) with the SHIPPABLE=YES attribute. In this case, the terminal definition is shipped to any application-owning region (AOR) when the terminal user invokes a transaction owned by (and defined to) that region. Definitions for advanced program-to-program communication (APPC) devices always have the SHIPPABLE=YES attribute set.

(The entry in the TOR could have been installed using CEDA INSTALL, the GRPLIST at system initialization, or autoinstall.)

The first time a transaction is invoked

For non-APPC devices (see Figure 3 on page 18), the following processing is performed:

- In the AOR, look for an existing skeleton TCTTE (TCTSK) whose REMOTENAME is the same as the local name in the TOR. If found, skip the following steps; otherwise:
- Issue ZC_INQUIRE to the TOR.
- In the TOR:
  - Send a builder parameter set (BPS) representing the TCTTE to the AOR.
  - Set on the SHIPPED flag (TCTEMROP) in the TCTTE.
  - Set on the SHIPPED flag (TCSEMROP) in the TCTSE for the AOR system.
  - Rewrite each entry to the catalog.
- In the AOR:
  - Use the existing name from the TOR.
  - INSTALL the terminal (DFHZATS does the remote install).
  - Set on the SHIPPED flag (TCTSKSHI) in the TCTSK.
  - Set on the SHIPPED flag (TCSEMROG) in the TCTSE for the TOR system.
  - Rewrite each entry to the catalog.

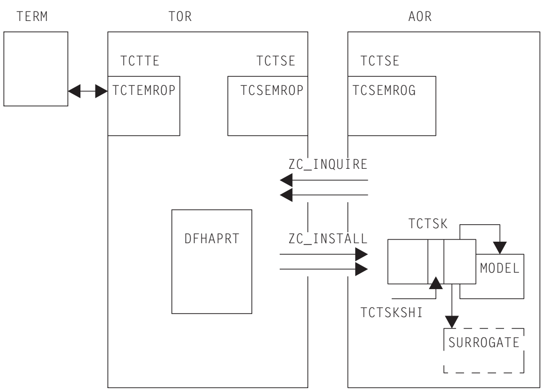


Figure 3. Transaction-routing flow for non-APPC devices

For APPC devices:

- In the AOR, look for an existing skeleton TCTTE (TCTSK) whose REMOTENAME is the same as the local name in the TOR. If found, skip the following steps; otherwise:
- INSTALL the terminal (DFHZATS does the remote install).
- Set on the SHIPPED flag (TCTSKSHI) in the TCTSK.
- Set on the SHIPPED flag (TCSEMROG) in the TCTSE for the TOR system.
- Rewrite each entry to the catalog.

When an autoinstalled TCTTE in a TOR is deleted

If this CICS is linked to a Pre CICS/ESA 4.1 system then the following occurs.

- If the deleted entry is flagged (TCTEMROP or TCSERDLR for APPC devices) as having been shipped, notify all remote systems that have received shipped definitions (TCSEMROP) that this terminal is being deleted.
- Determine from the TCTSK in the AOR whether a definition for this terminal has been shipped (TCTSKSHI). If so, call ZC_DELETE in the AOR.

If this CICS is linked to CICS/ESA 4.1 or above then relevant shipped terminals are deleted using a separate timing mechanism.

Modules

ZC (terminal control) together with the following:

Module	Function
DFHZATA	Autoinstall program
DFHZATA2	Console autoinstall program linkedits with DFHZATA
DFHZATD	Autoinstall delete program
DFHZATDX	Autoinstall control program
DFHZATDY	Sample autoinstall user exit
DFHZATR	Autoinstall restart program
DFHZATS	Remote autoinstall|delete program
DFHZCTRI	Trace interpretation for DFHZGAI
DFHZGAI	APPC-specific autoinstall functions

DFHZATDX

The DFHZATDX module provides user input to autoinstall processing. This module is a component of ZCP, and is the default autoinstall user program (that is, it is used if you choose not to provide your own). For further information about the DFHZATDX sample program, see the *CICS Customization Guide*.

DFHZATDX is also called when creating and deleting shipped terminals (skeletons).

DFHZATDY

DFHZATDY is a sample autoinstall user-replaceable program, which you must modify before you can use it. Its main function is to choose a template connection which is to be used in creating the new autoinstall connection clone. It also has to chose a name for the new connection. For further information about the DFHZATDY sample program see the *CICS Customization Guide*.

DFHZATDY is also called when creating and deleting shipped terminals (skeletons).

Diagnosing autoinstall problems

When diagnosing problems with autoinstall, consult the following list. If you have a problem with autoinstall of APPC devices, and the following list does not resolve the problem, see “Diagnosing APPC autoinstall problems” on page 20.

- The autoinstall model table (AMT) in an SDUMP
- CEMT INQUIRE AUTINSTMODEL—showing which models are installed
- TC level-1 trace, point ID AP FC8A—showing the CINIT RU contained in the AWE on entry to DFHZATA
- CADL, CSMT, and CSNE logs:
  - Autoinstall messages (DFHZC69xx)
  - Builder messages (DFHZC59xx, DFHZC62xx, and DFHZC63xx)
  - Terminal error messages
  - Information produced by DFHZNAC
- Dump taken in the user install program (the CICS-supplied default is DFHZATDX).

Most autoinstall problems can be grouped into three categories:

1. CICS rejects the LOGON request (message DFHZC2411 on the CSNE log).
2. The device rejects the actual BIND parameters (message DFHZC2403 on the CSNE log).
3. DFHZATA diagnoses a problem (message DFHZC69xx on the CADL log).

The first category of problem is caused by CICS being in the wrong state to accept an autoinstall, for example, CICS is shutting down or AUTOINSTALL is disabled (message DFHZC2433).

The second category of problem arises when the two BIND images match, but the BIND is rejected by the actual device (message DFHZC2403). For information about valid BIND parameters, consult the *3274 Control Unit Description and Programmer’s Guide*, GA23-0061.

The BIND image is contained in the CINIT RU passed to the LOGON exit. This is shown in trace point ID AP FC8A.

The reason for the third category of problem should be shown in the contents of the associated DFHZC69xx message on the CADL log. For example, message DFHZC6987 shows a BIND image mismatch between the incoming CINIT and the best available model (unlikely).

The length of each BIND image is found in the halfword preceding the image. A comparison is made for the ***smaller*** of the two length values, but not exceeding X'19' (decimal 25) bytes. The comparison is accomplished by an XC (exclusive OR) of the two BIND images into a work area. The result is ANDed with a mask that defines the required settings.

Additional bits are reset if the LU type, found in byte 14 of the BIND image, is 1, 2, 3, or 4. The final result in the work area must be 256 bytes of X'00'; any other value causes DFHZATA to reject the LOGON and write message DFHZC6987 to the CADL log.

For autoinstall to function correctly, three items must match:
1. The CINIT BIND image taken from the LOGMODE entry specified for the LU in the VTAMLST
2. The CICS MODEL BIND image built according to the specifications in the TYPETERM and TERMINAL entries
3. Device characteristics.

Diagnosing APPC autoinstall problems

When diagnosing APPC autoinstall problems, first refer to "Diagnosing autoinstall problems" on page 19. Most of points in that section apply to APPC autoinstall problems except for points that refer to autoinstall models.

Any APPC autoinstall problem should be accompanied by message DFHZC6920 to 23. These messages each have exception trace entries which should trace enough information to allow you to diagnose the problem.

There are three autoinstall instances of DFHZC2411:
- 4 System termination - CSASTIM tested.
- 5 VTAM termination - TCTVVTQS tested.
- 6 ISC=NO specified in the SIT.

There are two additional instances of DFHZC2433:
- 3 Autoinstall disabled - TCTVADEN tested in DFHZBLX.
- 4 Autoinstall temporarily disabled - TCTVADIN tested in DFHZBLX.

There are two additional instances of DFHZC3482:
- 3 No MVS storage for DFHZBLX to obtain MVS AWE storage.
- 4 No MVS storage for reporting a failure in a dummy work element.

Diagnosing console autoinstall problems

Much of the autoinstall for terminal advice is relevant. However, the following points should also be helpful.
1. Information about autoinstalled consoles is contained in:
   - The AWE (CWE)
   - The TCT prefix in the console BITMAP
   - The CCE
   - The SNEX
   - The interface to the autoinstall user-replaceable program.
2. When DFHZCNA is called with a modify command trace point AP FCF0 is issued and traces the CIB and CIB extension.
3. Trace point AP FCA7 shows the AWE/CWE created by DFHZCNA and passed to DFHZATA2.
4. DISCARD (used via CEMT or EXEC CICS) is useful whilst testing autoinstall for consoles.

5. CEMT INQUIRE TERMINAL is useful for seeing what consoles are installed and what their console names are.
6. The console names can vary depending on how the modify command was issued:
   - /f jobname,CEMT I TE from a TSO SDSF panel gives a console name of the USERID or the console name if changed using option 8 of SDSF.
   - f jobname,CEMT I TE from a TSO console gives a console name of the TSO USERID.
   - M/F jobname, CEMT I TE from the TSO SDSF panel gives a console name of MASTnn where nn is the names of the system. If SEC=YES is specified in the SIT then the user must first sign on with m/f jobname,CESN.
   - // MODIFY jobname,CEMT I TE from a job stream gives a console names of INTERNAL. If SEC=YES is specified in the SIT then the user must first sign on with m/f jobname,CESN.
7. The console name BITMAP is dumped in the TCP section of system dumps.
8. The extended control blocks are dumped if present when a system dump is taken.

VTAM exits

A VTAM exit is a special-purpose user-written routine that is scheduled by VTAM when the requested operation is complete. VTAM creates a trace record when the exit is given control.

RE entries represent RPL exits except SEND, RECEIVE, OPNDST, and CLSDST. UE entries represent non-RPL and asynchronous exits SCIP, LOGON, and LOSTERM.

See the *OS/390 eNetwork Communications Server: SNA Programming* manual, SC31-8573, for general VTAM exit information.

Trace

The following point IDs are provided for the autoinstall programs (DFHZATA, DFHZATD, DFHZATR, and DFHZATS), as part of terminal control:
- AP FC80 through AP FC8C, for which the trace levels are TC 1 and TC 2.

The following point IDs are provided for APPC autoinstall:
- AP FA00 to FA21, for which the trace levels are TC1 and TC2.

The following point IDs are provided for console autoinstall:
- AP FCF0
- AP FCA3 to FCA7

RE and UE trace points are recorded when the VTAM trace API option is requested by:

```
F NET,TRACE,TYPE=VTAM,OPTION=API,MODE=EXT
```

GTF must have been started with the USR option.

Each VTAM exit routine in CICS sets an ID byte in the TCTTE exit trace field (TCTEEIDA).

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 4. Autoinstall terminal model manager

The autoinstall terminal model manager (an OCO component of the AP domain) is responsible for managing all operations involving the autoinstall terminal model table. Autoinstall terminal models are used during the autoinstall logon process (see step 3 on page 12). They are installed either at system initialization or using CEDA INSTALL (see Chapter 42, "Resource definition online (RDO)," on page 343), and can be discarded using either the CEMT transaction or EXEC CICS commands.

The acronym AITM is often used for "autoinstall terminal model" in the contexts of both the manager and the associated table; it is also the name of one of the subroutine call formats.

The AITM manager is implemented as a set of subroutine interfaces.

Functions provided by the autoinstall terminal model manager

Table 2 summarizes the external subroutine interfaces provided by the autoinstall terminal model manager. It shows the subroutine call formats, the level-1 trace point IDs of the modules providing the functions for these formats, and the functions provided.

Table 2. Autoinstall terminal model manager's subroutine interfaces

Format	Trace	Function
AIIN	AP 0F10 AP 0F11	START_INIT COMPLETE_INIT
AIIQ	AP 0F18 AP 0F19	LOCATE_TERM_MODEL UNLOCK_TERM_MODEL INQUIRE_TERM_MODEL START_BROWSE GET_NEXT END_BROWSE
AITM	AP 0F08 AP 0F09	ADD_REPL_TERM_MODEL DELETE_TERM_MODEL

AIIN format, START_INIT function

The START_INIT function of the AIIN format is used to attach a CICS task to perform initialization of the AITM manager.

Input parameters

None.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

`OK|DISASTER|KERNERROR`

AIIN format, COMPLETE_INIT function

The COMPLETE_INIT function of the AIIN format is used to wait for the initialization task attached by the START_INIT function to complete processing.

Input parameters

None.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

`OK|DISASTER|KERNERROR`

AIIQ format, LOCATE_TERM_MODEL function

The LOCATE_TERM_MODEL function of the AIIQ format is used to obtain the attributes of a named autoinstall terminal model, and obtain a read lock on that entry in the AITM table in virtual storage.

Input parameters

TERM_MODEL_NAME
specifies the name of the autoinstall terminal model to be located.

BPS
identifies a buffer into which the attributes of the autoinstall terminal model are to be placed.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|KERNERROR`

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	TM_LOCATE_FAILED
EXCEPTION	TERM_MODEL_NOT_FOUND

AIIQ format, UNLOCK_TERM_MODEL function

The UNLOCK_TERM_MODEL function of the AIIQ format is used to release a read lock on a previously located entry from the AITM table in virtual storage.

Input parameters

TERM_MODEL_NAME
specifies the name of the autoinstall terminal model to be unlocked.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|KERNERROR`

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	TM_UNLOCK_FAILED
EXCEPTION	TERM_MODEL_NOT_FOUND

AIIQ format, INQUIRE_TERM_MODEL function

The INQUIRE_TERM_MODEL function of the AIIQ format is used to obtain the attributes of a named autoinstall terminal model. (No read lock is retained.)

Input parameters

TERM_MODEL_NAME
specifies the name of the autoinstall terminal model to be located.

BPS
identifies a buffer into which the attributes of the autoinstall terminal model are to be placed.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|KERNERROR`

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	TM_LOCATE_FAILED TM_UNLOCK_FAILED
EXCEPTION	TERM_MODEL_NOT_FOUND

AIIQ format, START_BROWSE function

The START_BROWSE function of the AIIQ format is used to initiate a browse of the AITM table. The browse starts at the beginning of the table.

Input parameters

None.

Output parameters

BROWSE_TOKEN
is a token used to refer to this browse session on subsequent browse requests.

RESPONSE
is the subroutine's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. It has this value:

START_BROWSE_FAILED

AIIQ format, GET_NEXT function

The GET_NEXT function of the AIIQ format is used to obtain the name and attributes of the next autoinstall terminal model in the AITM table for the specified browse session.

Input parameters

BROWSE_TOKEN
is the token identifying this browse session.

BPS
identifies a buffer to receive the attributes of the next entry in the AITM table.

Output parameters

TERM_MODEL_NAME
is the name of the next entry in the AITM table.

RESPONSE
is the subroutine's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	TM_GET_NEXT_FAILED TM_UNLOCK_FAILED
EXCEPTION	END_OF_MODELS

AIIQ format, END_BROWSE function

The END_BROWSE function of the AIIQ format is used to terminate a browse of the AITM table.

Input parameters

BROWSE_TOKEN
is the token identifying this browse session.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have either of these values:

OK|KERNERROR

AITM format, ADD_REPL_TERM_MODEL function

The ADD_REPL_TERM_MODEL function of the AITM format is used to add or update an entry in the AITM table in virtual storage, and record the entry on the CICS catalog.

Input parameters

TERM_MODEL_NAME
specifies the name of the autoinstall terminal model to be added or updated.

BPS
specifies the attributes of the named autoinstall terminal model.

SYSTEM_STATUS
specifies the status of the CICS system at the time of the call. It can have any one of these values:

```
COLD_START|WARM_START|ONLINE
```

where ONLINE means during execution.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|KERNERROR
```

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	NOT_INITIALISED ADD_REPL_FAILED
EXCEPTION	TERM_MODEL_IN_USE

AITM format, DELETE_TERM_MODEL function

The DELETE_TERM_MODEL function of the AITM format is used to remove an entry from the AITM table in virtual storage and the CICS catalog.

Input parameters

TERM_MODEL_NAME
specifies the name of the autoinstall terminal model to be added or updated.

SYSTEM_STATUS
specifies the status of the CICS system at the time of the call. It can have any one of these values:

```
COLD_START|WARM_START|ONLINE
```

where ONLINE means during execution.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|KERNERROR
```

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	NOT_INITIALISED DELETE_FAILED
EXCEPTION	TERM_MODEL_IN_USE TERM_MODEL_NOT_FOUND

Modules

Module	Function
DFHAIDUF	Formats the AITM manager control blocks in a CICS system dump
DFHAIIN1	Handles the following requests: START_INIT COMPLETE_INIT
DFHAIIN2	Runs as a CICS task to perform initialization of the AITM manager
DFHAIIQ	Handles the following requests: LOCATE_TERM_MODEL UNLOCK_TERM_MODEL INQUIRE_TERM_MODEL START_BROWSE GET_NEXT END_BROWSE
DFHAIRP	Initializes the AITM table at CICS startup
DFHAITM	Handles the following requests: ADD_REPL_TERM_MODEL DELETE_TERM_MODEL
DFHAPTRN	Interprets AITM manager trace entries

Exits

No global user exit points are provided for this component.

Trace

The following point IDs are provided for the AITM manager:
- AP 0F00 through AP 0F1F, for which the trace levels are AP 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 5. Basic mapping support

Basic mapping support (BMS) allows the CICS application programmer to have access to input and output data streams without including device-dependent code in the CICS application program.

BMS provides the following services:

Message routing
: This allows application programs to send output messages to one or more terminals not in direct control of the transaction.

Terminal paging
: This allows the user to prepare a multipage output message without regard to the physical size of the output terminal; the output can then be retrieved by page number in any order.

Device independence
: This allows the user to prepare output without regard to the control characters required for a terminal; CICS automatically inserts the control characters and eliminates trailing blanks from each line.

Most of the BMS programs are resident in the CICS nucleus.

Design overview

BMS is an interface between CICS and its application programs. BMS formats input and output display data in response to BMS commands in programs. To do this, it uses device information from CICS system tables, and formatting information from **maps** that you have prepared for the program.

BMS enables an application program to read in device-dependent data and convert it to a device-independent standard form, or to generate device-dependent output data from this device-independent standard form. In both cases, the structure of the device-independent standard form, and the layout of the data on the display terminal, are determined by a user-defined map. Related maps—for example, maps used in the same application program—are grouped together into a **map set**. See the *CICS Application Programming Guide* for further information about the definition and use of maps and map sets.

On some terminals (such as the IBM 8775 display terminal and the IBM 3290 information panel), the available display area may be divided into a set of related “logical” screens called **partitions**. The layout and properties of the set of partitions that can be simultaneously displayed on a terminal are defined by the BMS user in a **partition set**. See the *CICS Application Programming Guide* for further details about the definition and use of partition sets.

Maps, map sets, and partition sets are assembled offline using CICS macros. The user defines and names fields and groups of fields that can be written to and read from the devices supported by BMS. The assembled maps contain all the device-dependent control characters necessary for the proper manipulation of the data stream.

Associated with each map is a table of field names which is copied into each application program that uses the map. Data is passed to and from the application program under these field names. The application program is written to manipulate the data under the various field names so that alteration of a map format does not necessarily lead to changes in program logic. New fields can be added to a map format without making it necessary to reprogram existing applications.

Output data can be supplied from the application program by placing the data in the table under the appropriate field name. As an alternative, output maps can contain field default data that is sent when data is not supplied by an application program. This facility permits the specification of titles, headers, and so on, for output maps.

Optionally, the display of all the default data can be suppressed by the application program for any output map. Each time a map is used, the application program can temporarily modify the attributes of any named field in the output map. The extended attributes can also be modified if maps are defined with the DSATTS operand.

Output map fields with no field names can contain default data, but the application program cannot replace the default data or modify the attributes of unnamed fields.

For input, the user assembles a map defining the fields that can be written to and received from a particular device. Any data received for a particular field is moved across using the field name in the symbolic storage definition for the map. Light-pen-detectable fields defined in an input map are flagged as detected if present in an IBM 3270 Information Display System input stream. An input map for a particular case can specify a subset of the fields potentially receivable; any fields received and not represented in that map are discarded. This permits the number of fields from a map that can be typed or selected to be changed, without making it necessary to reprogram applications that currently receive data from the map.

Maps are stored in the CICS program load library. When a map is required by BMS, a copy is automatically retrieved by CICS from the program load library without application program action. Multiple users of a map contained in the program load library share a single copy in main storage.

BMS permits any valid combination of field attributes to be specified by the user when generating maps. Inclusion of BMS in CICS is a system generation option and does not prevent the application program from accessing a particular device in native mode (without using BMS). Intermixing BMS and native mode support for a terminal from the same application program may yield unpredictable results. When using mixed mode support, it is the user’s responsibility to ensure the correct construction and interpretation of native mode data streams.

BMS permits the application program to pass a native mode data stream that has already been read in, and (if, for a terminal of the IBM 3270 Information Display System, the screen has been formatted) to interpret this data stream according to a given input map. This facility allows data entered with the initial reading of a transaction to be successfully mapped using BMS.

BMS provides the following services:
- Message routing
- Terminal paging
- Device independence.

Message routing

Message routing permits the application program to send an output message to one or more terminals not in direct control of the transaction. The message is automatically sent to a terminal if the terminal status allows reception of the message. If a terminal is not immediately eligible to receive the message, the message is preserved for that terminal until a change in terminal status allows it to be sent. The message routing function is used by the CICS message-switching transaction.

A BMS map that specifies extended attributes can be used for terminals that do not support extended attributes. When sending data to a variety of terminals, some of the terminals may support extended attributes and others may not. When a BMS ROUTE request is processed, BMS looks at the TCTTEs for all specified terminals and constructs a set of all the supported attributes.

A data stream is produced by BMS using this set of attributes, and the data stream and set of attributes for each page are written to a temporary-storage record. When the page is later read from temporary storage, the data stream for each terminal is modified, if necessary, to delete attributes not supported by that terminal.

Terminal paging

Terminal paging allows the user to prepare more output than can be conveniently or physically displayed at the receiving terminal. The output can then be retrieved by pages in any order; that is, in the order in which they were prepared or by skipping forward or backward in the output pages.

Terminal paging also provides the ability to combine several small areas into one area, which is then sent to the terminal. This enables the user to prepare output without regard for the record size imposed by the output terminal.

CICS provides the terminal operator with a generalized page retrieval facility that can be used to retrieve and dispose of pages.

Device independence

Device independence allows the user to prepare output without regard for the control characters required for message heading, line separation, and so on. Input to device independence consists of a data string with optional new-line characters.

Device independence divides the data string into lines no longer than those defined for the particular terminal. If new-line characters appear occasionally in the data string to further define line lengths, they are not ignored. CICS inserts the appropriate leading characters, carriage returns, and idle characters, and eliminates trailing blanks from each line. If the device does not support extended attributes, the extended attributes are ignored.

CICS allows the user to set horizontal and vertical tabs on those devices that support the facility (for example, the IBM 3767 Communication Terminal, and the IBM 3770 Data Communication System). For such devices, CICS supports data compression inbound and data compression outbound, based on the tab characteristics in the data stream under the control of the appropriate maps.

Control blocks

BMS makes use of the following control blocks (see Figure 4 on page 33):

DSECT	Function
DFHMAPDS	Defines a physical map. It contains overlays for map set data, map data, and field data. The physical map set is stored in the CICS program library and requires a resource definition when loaded into main storage by BMS.
DFHMCAD	Defines a mapping control area (MCA). MCAs are used in DFHM32 and DFHML1 to merge (both) and sort (DFHML1 only) fields in different maps in the chain of map copies. The MCA contains field position, flags, and pointers to map and application data structure relating to this field.
DFHMCBDS	Defines the message control block (MCB). MCBs are built and referenced by DFHTPR. There is one MCB per level of page chaining. The MCBs are chained together, with the head of the chain anchored off the TCTTE BMS extension. The MCB contains a copy of the MCR, with additional working data.
DFHMCRDS	Defines the message control record (MCR). MCRs are held in CICS temporary storage. There is one MCR per BMS message in temporary storage. The MCR contains data such as the number of pages in this message, the list of target terminals for this message, data on which pages are for which LDCs or partitions, and so on. The MCR is written to temporary storage by DFHMCP. It is read and purged by DFHTPR, DFHTPS, and DFHTPQ.

DSECT	Function
DFHOSPWA	Defines the output services processor work area (OSPWA). This is the main BMS control block. For standard and full-function BMS, there is an OSPWA that is chained off the TCA and is built by DFHMCP on the first BMS command in a transaction. It contains a copy of the BMS TCA request bytes, together with the BMS status and working area. DFHTPR has its own private OSPWA. This overlays the TWA for DFHTPR unless SEND PAGE RETAIN is used. If SEND PAGE RETAIN is used, DFHTPR obtains an additional OSPWA, and chains the base OSPWA off the new OSPWA. This avoids DFHTPR damaging the base OSPWA. The OSPWA is deleted during task termination. A shorter version of the OSPWA is used by DFHMCPE (part of both the minimum-function BMS mapping control program DFHMCPE$ and also the BMS fast-path module DFHMCX). It is built in DFHMCPE's LIFO storage, and includes space for the request information from the TCA. The DFHMCPE OSPWA is defined within DFHMCPE.
DFHPGADS	Defines a page control area (PGA). DFHTPP builds a PGA at the end of the device data stream in the terminal input/output area (TIOA) (addressed as ADDR(TIOADBA) + TIOATDL) for the SET and PAGING disposition. The PGA contains the 3270 write control character (WCC), flags about the type of TC write required, and the extended features used in this page of data stream.
DFHPSDDS	Defines a physical partition set. The partition set is stored in the CICS program library and requires a resource definition when loaded into main storage by BMS.
DFHTTPDS	Defines the terminal type parameter (TTP). This contains information for a terminal type. Note that BMS builds pages on a TTP basis. For standard and full-function BMS, DFHRLR builds TTPs as follows: 1. A "direct TTP" is built for the transaction terminal. If this supports partitions or LDCs, a further direct TTP is built for each referenced LDC or partition. This contains data for that LDC or partition. These direct TTPs are chained together, and the head of the chain is contained in the OSPWA. Direct TTPs are deleted by DFHMCP on a SEND PAGE, PURGE MESSAGE, or SEND PARTNSET command. 2. If routing is in effect, there is a chain of routed TTPs, with one TTP per terminal type in the route list. Routed TTPs are deleted by DFHMCP on a SEND PAGE or PURGE MESSAGE command. Most of BMS uses the TTP rather than the TCTTE to determine terminal-related information.
TCTTETTE	The TCTTETTE DSECT in the DFHTCTZE macro defines the TCTTE BMS extension. It is chained off the TCTTE (TCTTETEA field).
DFHTPE	Defines the BMS partition extension. This is chained off the TCTTE BMS extension if the terminal supports partitions.

See the *CICS Data Areas* manual for a detailed description of these control blocks.

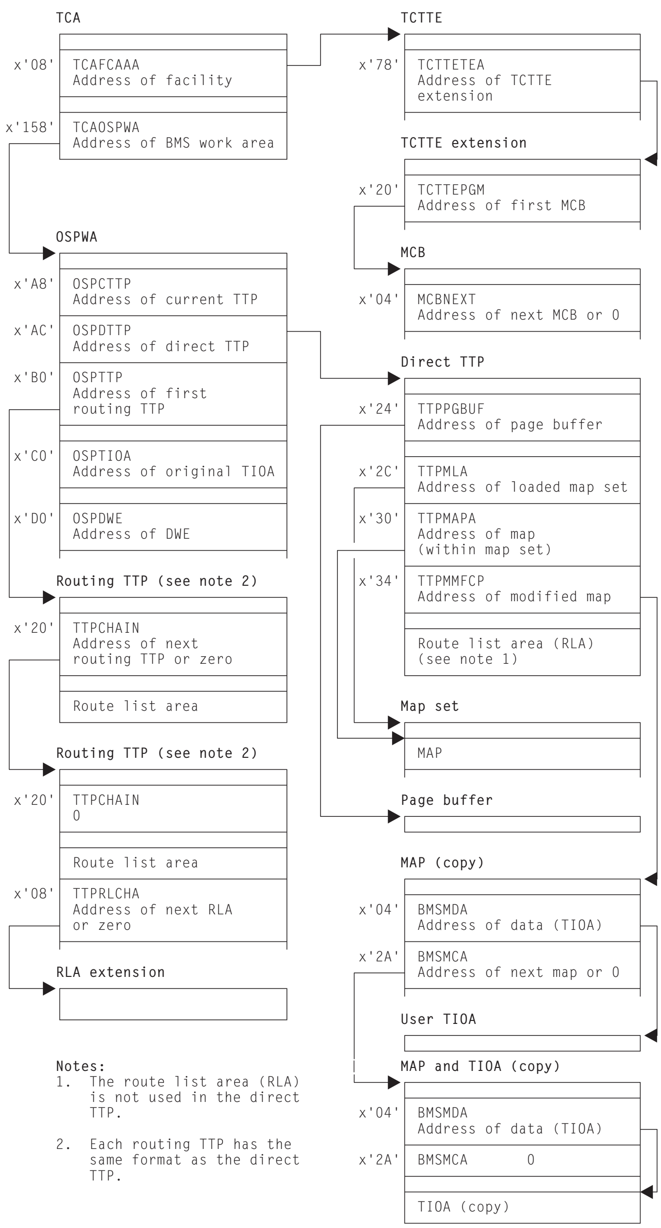


Figure 4. Control blocks associated with basic mapping support (BMS)

Modules

BMS makes use of the following modules (see Figure 5 on page 35):

Module	Function
DFHDSB	Addresses the page buffer, which was composed by the page and text build program (DFHPBP).
DFHEMS	The EXEC interface processor for BMS commands.
DFHIIP	Called in response to requests for BMS services involving terminals other than IBM 3270 Information Display Systems.

Basic mapping support

Module	Function
DFHMCP	The interface between application programs and the modules that perform mapping, message switching, page and text building, device-dependent output preparation, and message disposition to terminals, temporary-storage areas, or the application program.
DFHMCX	The BMS fast path module for standard and full-function BMS, and the program for minimum BMS support. It is called by DFHMCP if the request satisfies one of the following conditions: • It is a non-cumulative direct terminal send map or receive map issued by a command-level program. • It is for a 3270 display or an LU3 printer which does not support outboard formatting. If the terminal supports partitions, it is in the base state. • The CSPQ transaction has been started. • The message disposition has not changed.
DFHM32	Called in response to requests for BMS services involving terminals of the 3270 Information Display System.
DFHPBP	Processes all BMS output requests (SEND MAP, SEND PAGE, and SEND TEXT). It performs the following functions: • Positions the data in the page, either by actually placing it in a buffer, or by copying it and adjusting the map for an IBM 3270 Information Display System (SEND MAP ACCUM) • Places the data into the page buffer (SEND TEXT ACCUM) • Inserts device-dependent control characters for other than 3270 Information Display System devices, removing extended attributes.
DFHPHP	Processes terminal operations that involve partitions.
DFHRLR	Builds terminal type parameters (TTPs), which are the main blocks for building and writing out data in BMS.
DFHTPP	Directs completed pages to a destination specified in the BMS output request: SEND TEXT sends to the originating terminal; SEND MAP PAGING or SEND TEXT PAGING directs to temporary storage; and SEND MAP SET or SEND TEXT SET directs to a list of completed pages that are returned to the application program).
DFHTPQ	Checks the chain of automatic initiate descriptors (AIDs) to detect and delete AIDs that have been on the chain for an interval exceeding the purge delay time interval specified by the PRGDLAY system initialization parameter, if this has a nonzero value.
DFHTPR	Processes messages built by BMS and placed in temporary storage.
DFHTPS	Invoked for each terminal type to which a BMS logical message built with SEND MAP PAGING or SEND TEXT PAGING is to be sent. For each terminal designated by the originating application program, DFHTPR is scheduled to display the first page of the logical message if the terminal is in paging status, or the complete message if it is in autopage status.

Basic mapping support (BMS) is provided by means of a number of modules, each of which interfaces with other BMS modules, CICS control components, and application programs. The maps that are handled by BMS may be new maps, created to utilize BMS mapping capabilities. The interrelationships of CICS programs requesting mapping services are summarized in Figure 5 on page 35. Further details for specific programs within BMS are given on pages that follow.

One of three versions (MINIMUM, STANDARD, or FULL) of basic mapping support can be selected by the system initialization parameter BMS (see the *CICS System Definition Guide*). Where the generated versions of a BMS module differ according to the level of function provided, a suffix identifies the version as follows:
- E$ for minimum function
- A$ for standard function
- 1$ for full function.

In the module lists that follow, an asterisk (*) after a module name shows that the module is suffixed in this way. Elsewhere in this book, however, the BMS modules are usually referenced by their unsuffixed names with no distinction made between the minimum, standard, and full-function versions.

The module used by all three versions of BMS (minimum, standard, and full-function) is:
- DFHMCP* (mapping control program).

Additional modules used by both standard and full-function versions of BMS are:
- DFHDSB* (data stream build)
- DFHIIP* (non-3270 input mapping)
- DFHMCX (fast path module)
- DFHML1 (LU1 printer mapping)
- DFHM32* (3270 mapping)
- DFHPBP* (page build program)
- DFHPHP (partition handling program)
- DFHRLR* (route list resolution)
- DFHTPP* (terminal page processor).

Additional modules used only by full-function BMS are:
- DFHTPQ (terminal page cleanup)
- DFHTPR (terminal page retrieval)
- DFHTPS (terminal page scheduling).

A detailed description of each of these modules follows in alphabetic order of module name.

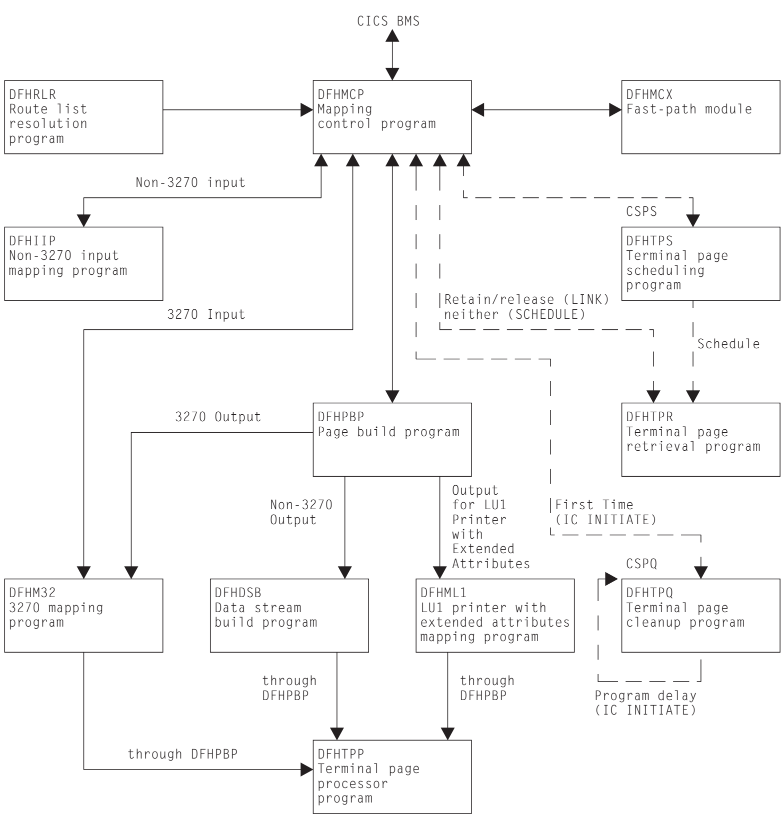


Figure 5. Modules associated with basic mapping support (BMS)

DFHDSB (data stream build)

The data stream build program addresses the page buffer, composed by the page and text build program (DFHPBP). The page buffer contains lines of output data that are to be written to a terminal other than an IBM 3270 Information Display System. The number of lines is contained in the TTPLINES field. The data stream build program performs the following functions on the data in the page buffer:

- Truncates trailing blanks within data lines
- Substitutes strings of physical device control characters for logical new-line characters that terminate each line of data
- Provides a format management header (FMH) for some VTAM-supported devices
- Allows horizontal and vertical tab processing.

Figure 6 shows the relationships between the components of data stream build.

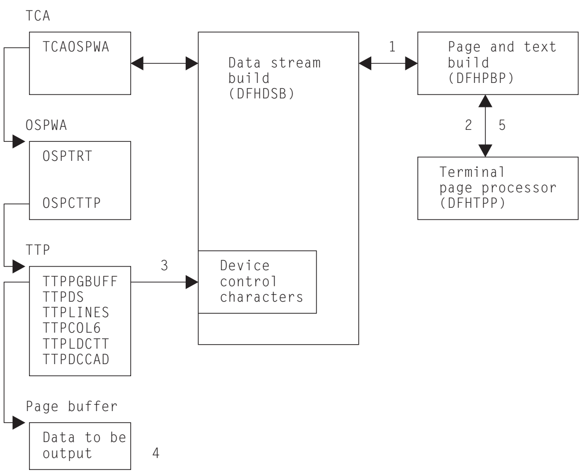


Figure 6. Data stream build interfaces

Notes:

1. DFHDSB is entered from the page build program to process the page buffer.
2. For SEND TEXT commands with the NOEDIT option specified, page buffer compression is skipped and control returns to DFHPBP, which calls the terminal page processor (DFHTPP).
3. For SEND TEXT commands without the NOEDIT option, the appropriate device control characters for the target device are selected for substitution.
4. The page buffer containing the data to be compressed is located through the address stored at TTPPGBUF.
5. After compression of the page buffer data, control returns to DFHPBP, which calls DFHTPP to provide disposition of the page.

DFHIIP (non-3270 input mapping)

The non-3270 input mapping program (DFHIIP) is called in response to requests for BMS services involving terminals other than IBM 3270 Information Display Systems.

Figure 7 on page 37 shows the relationships between the components of non-3270 input mapping.

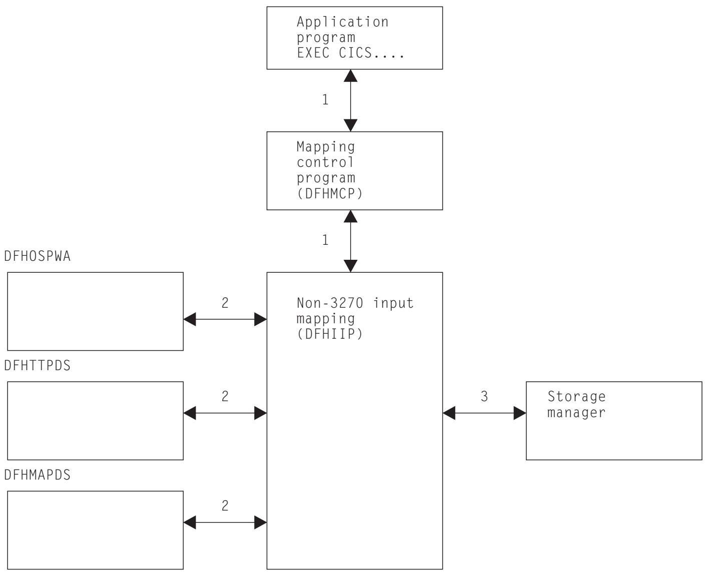


Figure 7. Non-3270 input mapping interfaces

Notes:

1. A RECEIVE MAP request by an application program, communicating with other than an IBM 3270 Information Display System, passes information through the TCA through the mapping control program (DFHMCP) to DFHIIP.
2. The map required for an operation is either passed by the application program or loaded by DFHMCP.
3. DFHIIP communicates with storage control to obtain and release buffers for mapping operations.

DFHMCP (mapping control program)

The mapping control program (DFHMCP) is the interface between application programs and the modules that perform mapping, message switching, page and text building, device-dependent output preparation, and message disposition to terminals, temporary-storage areas, or the application program.

Figure 8 on page 38 shows the relationships between the components of mapping control.

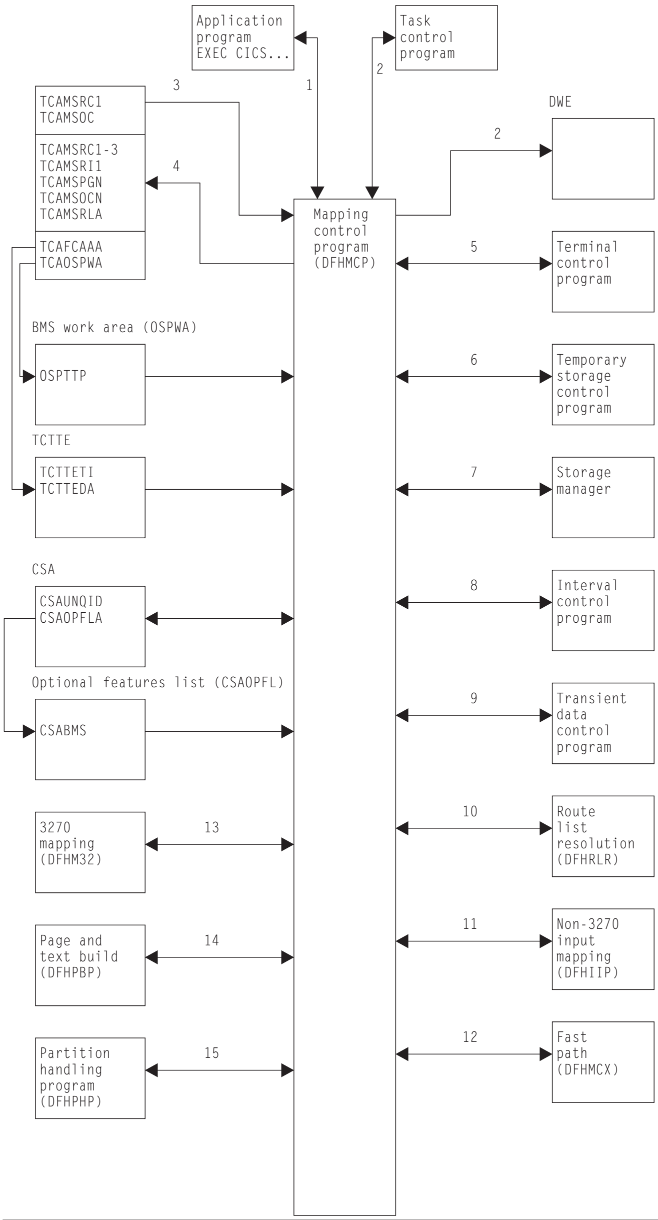


Figure 8. Mapping control program interfaces

Notes:

1. This program is entered when an application program issues a request for basic mapping support services.
2. It may also be called by task control to process a deferred work element (DWE) if an application program terminates and there are partial pages in storage, or the message control record (MCR) created during execution of the task has not been placed in temporary storage.
3. The following information is returned to the requester: error codes, page overflow information, and (for a SEND MAP SET or SEND TEXT SET command) a list of completed pages.
4. DFHMCP communicates with temporary storage control to put the MCR for routed or stored messages, if a ROUTE command, or SEND MAP PAGING or SEND TEXT PAGING command is issued. A DELETEQ TS command is issued to request that a message be purged from temporary storage if a PURGE MESSAGE command is issued.
5. DFHMCP communicates with storage control to:

- Acquire and free storage in which the MCR is built (a SEND MAP command after a SEND MAP PAGING, SEND TEXT PAGING, or ROUTE command)
- Acquire and free storage in which to copy the message title (a ROUTE command with the TITLE option specified)
- Acquire storage to build automatic initiate descriptors (AIDs) for non-routed messages, or routed messages to be delivered immediately (a SEND PAGE command)
- Acquire a BMS work area (OSPWA) at the time of the initial BMS request
- Acquire and free an area used for user request data if a SEND PAGE command must be simulated before processing the user's request
- Free the returned page list (a DELETEQ TS command)
- Free map copies if SEND PAGE command was issued and pages were being built in response to SEND PAGE commands
- Free terminal type parameters (TTPs) (SEND PAGE command).

6. DFHMCP communicates with program manager to:
   - Load and delete map sets
   - Link to the terminal page retrieval program (DFHTPR) to process one or more pages of a message if a SEND PAGE command is issued with the RETAIN or RELEASE option specified
   - Abnormally terminate tasks that incur errors that cannot be corrected.
7. DFHMCP communicates with interval control to:
   - Initiate transaction CSPQ
   - Obtain the current time of day, which is then used to time stamp AIDs for routed messages
   - Initiate transaction CSPS for messages to be delivered later.
8. DFHMCP communicates with task control to schedule transaction CSPQ for every terminal that is to receive a routed message to be delivered immediately.
9. Transient data control is used to send error and information messages to the master terminal.
10. Route list resolution (DFHRLR) is used to collect terminals from a user-supplied route list or from the entire TCT by terminal type, and build a terminal type parameter (TTP), which controls message building, for each terminal type. It is also used to build a single-element TTP for the originating terminal.
11. Non-3270 input mapping (DFHIIP) is used to process RECEIVE MAP requests for a terminal other than an IBM 3270 Information Display System.
12. The mapping control program calls DFHMCX if the request is eligible for the BMS fast-path module.
13. 3270 mapping (DFHM32) is used to process RECEIVE MAP requests for an IBM 3270 Information Display System.
14. Page and text build (DFHPBP) processes the following output requests:
15. Page and text build program (DFHPBP) processes all BMS output requests
    SEND MAP
    SEND MAP PAGING
    SEND MAP SET
    SEND PAGE
    SEND TEXT
    SEND TEXT PAGING
    SEND TEXT SET.

    For 3270 output, DFHM32 is called; for other output, DFHML1 is called.
16. The partition handling program (DFHPHP) is called when the data is in an inbound structured field. DFHPHP extracts the partition ID, device AID, and cursor address.

DFHML1 (LU1 printer with extended attributes mapping)

The LU1 printer with extended attributes mapping program, DFHML1, is called in response to requests for BMS services involving terminals of the 3270 Information Display System. Figure 9 shows how the DFHML1 program responds to these requests.

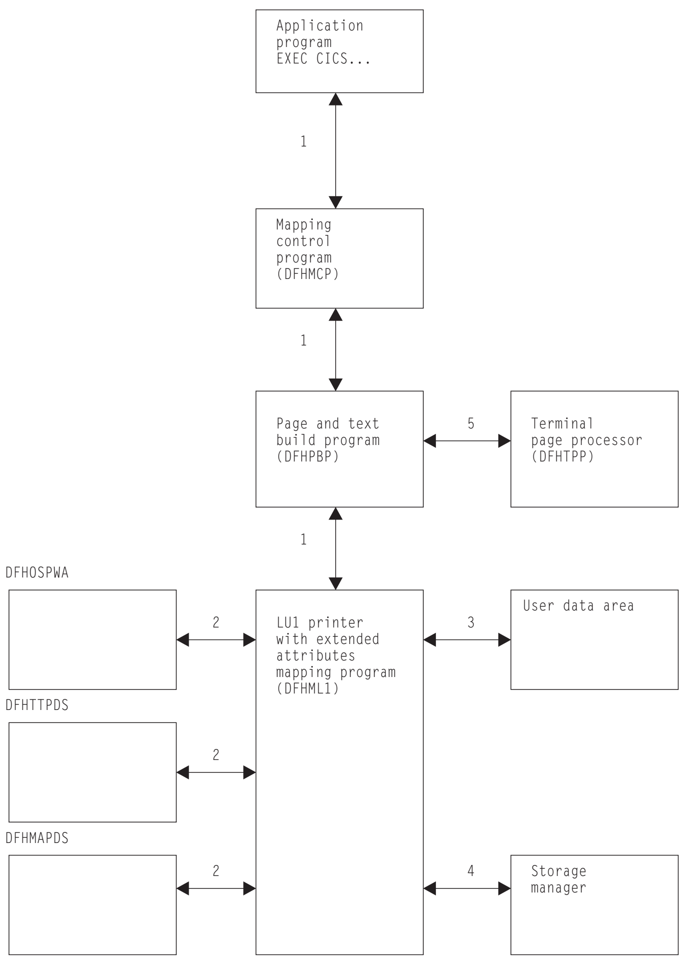


Figure 9. LU1 printer with extended attributes mapping program interfaces

Notes:

1. The following types of requests, by application programs communicating with LU1 printer mapping, pass information through the mapping control program (DFHMCP), and the page and text build program (DFHPBP), to DFHML1:
   SEND MAP ACCUM
   SEND MAP SET
   SEND TEXT
   SEND TEXT ACCUM
   SEND TEXT SET

   For one page of output, DFHML1 acquires an area and formats it into a chain of control blocks known as map control areas (MCAs). Each MCA corresponds to one map on the page and contains information about chaining down the maps and processing the fields in each map. DFHML1 then builds the data stream directly from the maps and the TIOAs.
2. Maps are either passed by the application program or loaded by DFHMCP.
3. The address of a terminal input/output area (TIOA) is supplied by the application program for all requests.

4. DFHML1 communicates with storage control to obtain and release storage for MCAs and for the mapped data.
5. All requests (see note 1 on page 40) are sent to a designated destination by the terminal page processor (DFHTPP), after the return of control to DFHPBP.

DFHM32 (3270 mapping)

The 3270 mapping program (DFHM32) is called in response to requests for BMS services involving terminals of the 3270 Information Display System. Figure 10 shows how the 3270 mapping program responds to these requests.

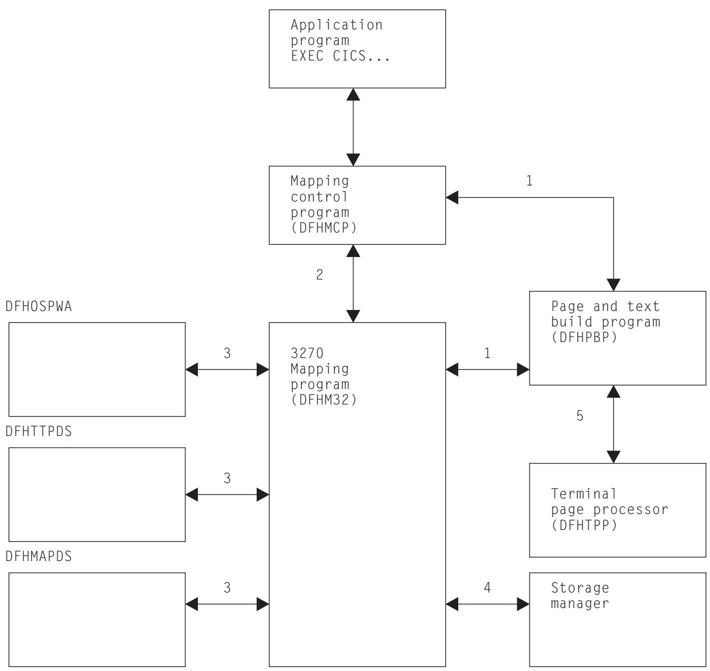


Figure 10. 3270 mapping program interfaces

Notes:

1. The following types of requests by an application program communicating with an IBM 3270 Information Display System passes information through the TCA by way of the mapping control program (DFHMCP) and the page and text build program (DFHPBP) to DFHM32:
   SEND MAP ACCUM
   SEND MAP PAGING
   SEND MAP SET
   SEND TEXT
   SEND TEXT ACCUM
   SEND TEXT PAGING
   SEND TEXT SET

   For one page of output, DFHM32 acquires an area and formats it into a chain of control blocks known as map control areas (MCAs). Each MCA corresponds to one map on the page and contains information for chaining down the maps and processing the fields in each map. DFHM32 then builds the data stream directly from the maps and the TIOAs.
2. A RECEIVE MAP or RECEIVE MAP FROM request by an application program communicating with an IBM 3270 Information Display System passes information through the TCA through the message control program (DFHMCP) to DFHM32.
3. Maps are either passed by the application program or loaded by DFHMCP.

4. DFHM32 communicates with storage control to obtain and release storage for MCAs and for the mapped data.
5. All output requests (see note 1 on page 41) are sent to a designated destination by the terminal page processor (DFHTPP) after control is returned to DFHPBP.

DFHPBP (page and text build)

The page and text build program (DFHPBP) processes all BMS output requests

SEND MAP
SEND MAP PAGING
SEND MAP SET
SEND PAGE
SEND TEXT
SEND TEXT PAGING
SEND TEXT SET.

It performs the following functions:

- Positions the data in the page, either by actually placing it in a buffer, or by copying it and adjusting the map for an IBM 3270 Information Display System (SEND MAP ACCUM)
- Places the data into the page buffer (SEND TEXT ACCUM)
- Inserts device-dependent control characters for other than 3270 Information Display System devices, removing extended attributes.

Figure 11 on page 43 shows the relationships between the components of page and text build.

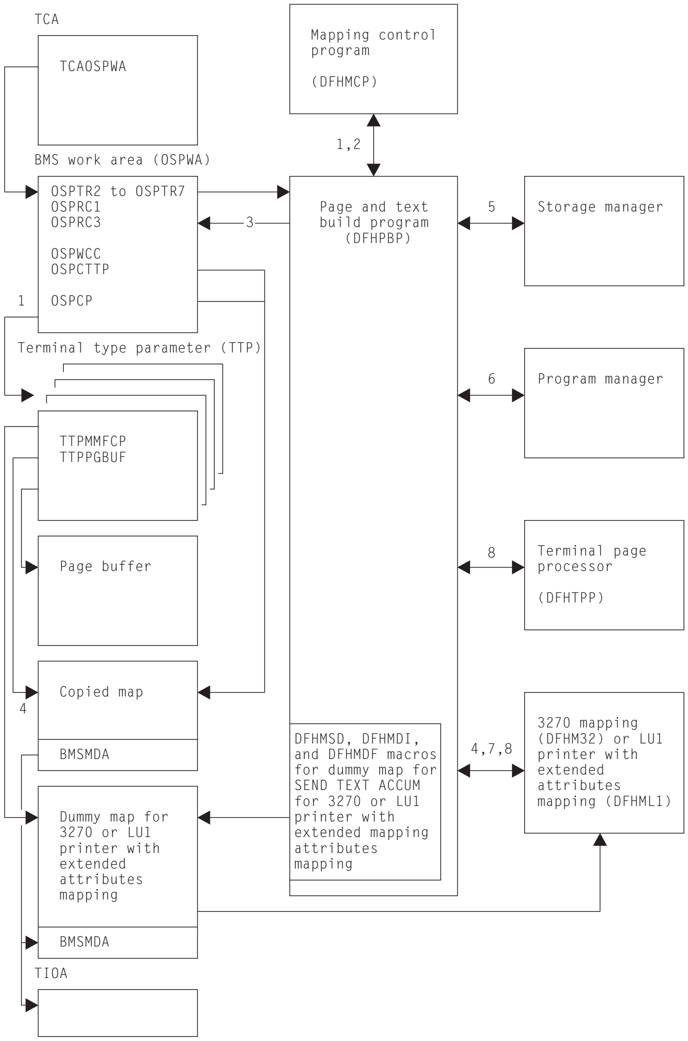


Figure 11. Page and text build program interfaces

Notes:

1. DFHPBP is entered from the mapping control program, DFHMCP, to process all BMS output requests. It is called once for each terminal type parameter (TTP) on the TTP chain pointed to by OSPTTP. The current TTP in the chain is pointed to by OSPCTTP.
2. DFHPBP returns control to DFHMCP when request processing is complete, or when the page must be written out before a SEND MAP ACCUM request can be processed and an OFLOW=symbolic address operand was specified.
3. OSPTR2, OSPTR3, ..., OSPTR7 contain request data from the DFHBMS macro expansion. OSPRC1 and OSPRC3 contain return codes to be examined by DFHMCP.
4. For a SEND MAP ACCUM request for an IBM 3270 Information Display System, the map is copied and chained to the TTP. For a SEND TEXT ACCUM request for an IBM 3270 Information Display System, a dummy map is created and chained to the TTP. When a page is complete, control is given to 3270 mapping (DFHM32), which combines the map copies chained to the TTP and maps the data.

   For a SEND MAP ACCUM request for an LU1 printer with extended attributes, the map is copied and chained to the TTP. For a SEND TEXT ACCUM request, a dummy map is created and chained to the TTP. When a page is complete, control is given to the LU1 printer mapping program (DFHML1), which combines the map copies chained to the TTP and maps the data.
5. DFHPBP communicates with storage control to:

- Acquire and free buffers in which pages are built
- Acquire storage for copies of maps for SEND MAP ACCUM or SEND TEXT ACCUM
- Acquire storage for a copy of the user's data for SEND MAP ACCUM or SEND TEXT ACCUM.

6. DFHPBP requests program manager to terminate a transaction abnormally (ABEND) if certain errors occur that cannot be corrected.
7. A SEND TEXT ACCUM request for an IBM 3270 Information Display System causes a map set consisting of one dummy map to be passed to 3270 mapping (DFHM32). The map has one field with attributes FREEKB and FRSET.

   SEND TEXT ACCUM requests for an LU1 printer cause a map set consisting of one dummy map to be passed to the LU1 printer mapping program (DFHML1). The map has one field with attributes FREEKB and FRSET.
8. If the page is being constructed for an IBM 3270 Information Display System, control is given to DFHM32 to map the data and then to DFHTPP to output the page.

   If the page is being constructed for an LU1 printer, control is given to DFHML1 to map the data, and then to DFHTPP to output the page. Otherwise, control is given to DFHDSB to add device dependencies to the page, and then to the terminal page processor (DFHTPP) to output the page.

DFHPHP (partition handling program)

The partition handling program (DFHPHP) processes terminal operations that involve partitions. DFHPHP has one entry point, and starts with a branch table that passes control to the required routine according to the request. It consists of routines that perform the following functions:

- PHPPSI tests whether there is a partition set in storage. If there is and it is not the required partition set, that partition set is deleted. When no partition set is in storage, an attempt is made to load the appropriate partition set.
- PHPPSC builds a data stream to destroy any partitions that may already be loaded on the terminal, creates the partition set designated by the application partition set, and sets the name of the partition set in the TCTTE to be the name of the application partition set.
- PHPPIN extracts the AID, cursor address, and partition ID. The AID and cursor address are put in the TCTTE, and the partition ID is converted to a partition name and returned to the caller. A check is made that the partition ID is a member of the application partition set.
- PHPPXE sends a data stream to a terminal to activate the appropriate partition and sends an error message to any error message partition if input arrived from an unexpected partition.

Figure 12 on page 45 shows the relationships between the components of partition handling.

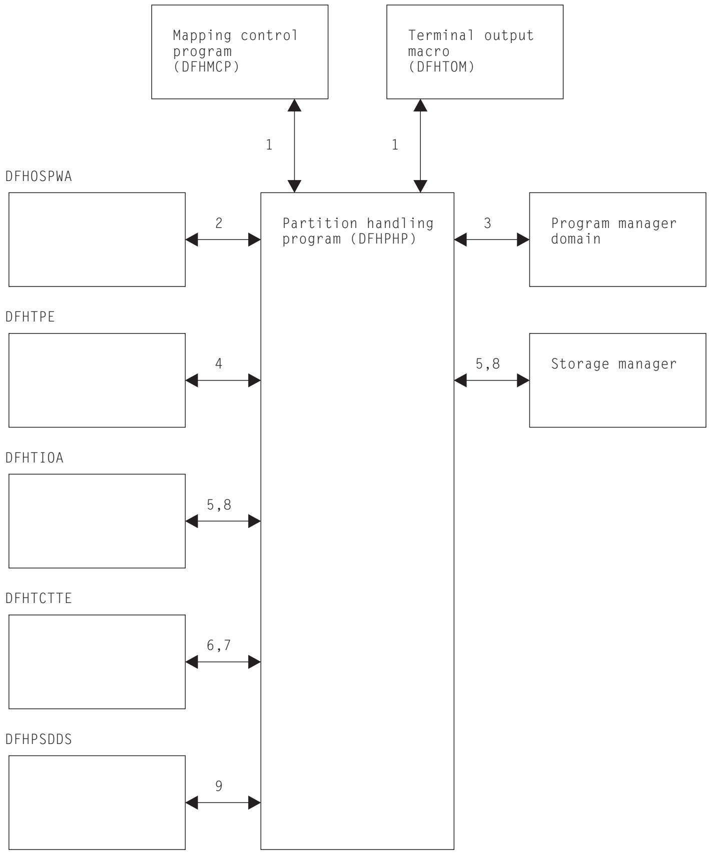


Figure 12. Partition handling program interfaces

Notes:

1. DFHPHP is called by the mapping control program (DFHMCP) and by the terminal output macro (DFHTOM).
2. PHPPSI refers to OSPWA to check whether a partition set is loaded.
3. PHPPSI communicates with program manager to load the partition set.
4. PHPPSI puts the name of the partition set in TPE (terminal partition extension) as the application partition set.
5. PHPPSC calls storage control to acquire a TIOA in which to build and free the original TIOA.
6. PHPPSC sets a slot in the TCTTE to be the partition set data stream concatenated with the terminal partition set name if the terminal is not in the base state.
7. PHPPIN places the AID and the cursor address in the TCTTE.
8. PHPPXE calls storage control to get a TIOA, retrieves the error message text by calling the message domain, fills the TIOA with data, transmits the data, and frees the TIOA.
9. PHPPSC references the partition set object to build the partition creation data stream.

DFHRLR (route list resolution program)

The route list resolution program (DFHRLR) builds terminal type parameters (TTPs), which are the main blocks for building and writing out data in BMS.

Figure 13 on page 46 shows the route list resolution program interfaces.

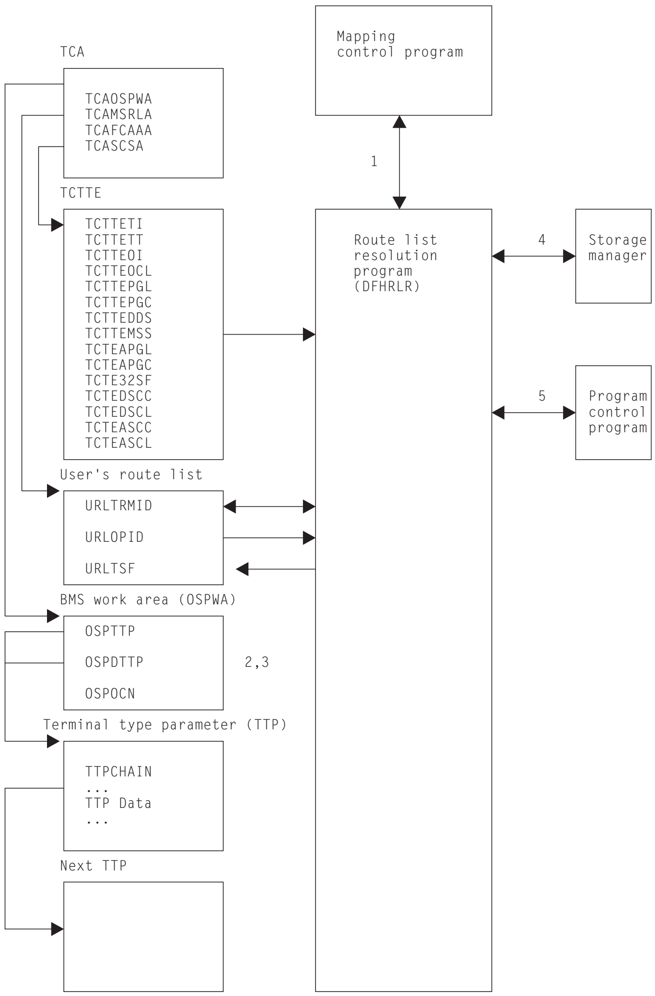


Figure 13. Route list resolution program interfaces

Notes:

1. DFHRLR is called by the mapping control program (DFHMCP) to determine the grouping of terminal destinations.
2. If data is to be routed, DFHRLR groups the terminals in the user's route list by terminal type and builds a routing TTP for each type. For each TTP, the supported attributes of the corresponding terminals are accumulated. The address of the first routing TTP in the chain of TTPs is placed in OSPTTP.
3. If data is not to be routed, a direct TTP is built for the originating terminal and its address is placed in OSPDTTP.
4. DFHRLR communicates with storage control to acquire storage for the TTP.
5. Program manager services are requested by means of an ABEND command if errors occur that cannot be corrected.

DFHTPP (terminal page processor)

The terminal page processor (DFHTPP) directs completed pages to a destination specified in the BMS output request:

- SEND MAP or SEND TEXT sends to the originating terminal
- SEND MAP PAGING or SEND TEXT PAGING directs to temporary storage
- SEND MAP SET or SEND TEXT SET directs to a list of completed pages that are returned to the application program.

Figure 14 shows the relationships between the terminal page processor and other components in response to BMS output requests.

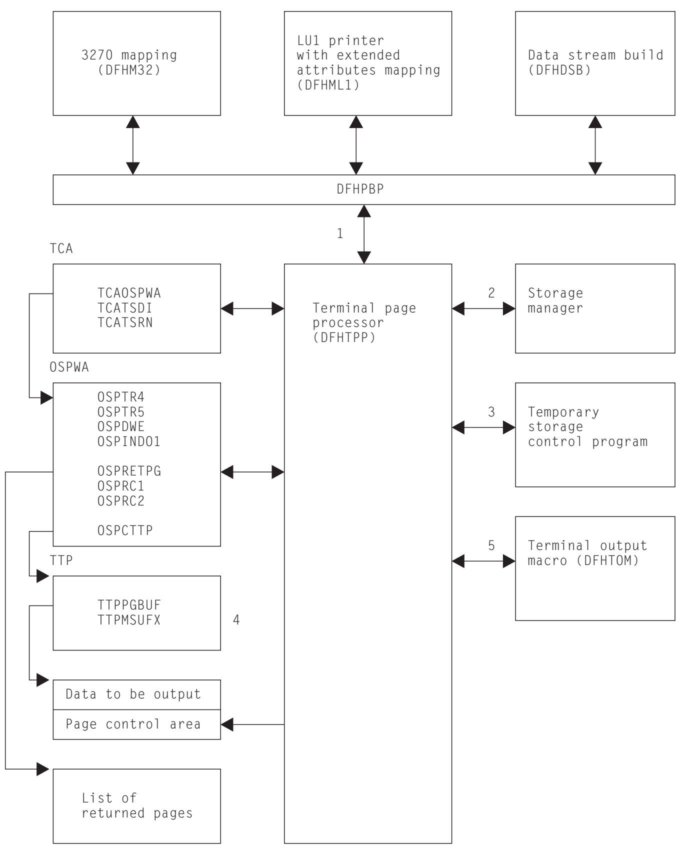


Figure 14. Terminal page processor interfaces

Notes:

1. DFHTPP is entered from DFHPBP after processing by 3270 mapping (DFHM32) for 3270s, by LU1 printer with extended attributes mapping (DFHML1) for those LU1 printers, and by data stream build (DFHDSB) for other devices.
2. DFHTPP communicates with storage control to obtain:
   - The return list (to store the address of completed pages to be returned to the program)
   - Deferred work elements (DWEs), which ensure that message control information is written to disk, even if the program neglects to issue a SEND PAGE request
   - Storage for a list that correlates pages on temporary storage with the logical device codes for which they are destined.
3. Temporary-storage control is used to store pages and the message control record (MCR) for messages stored on temporary storage.
4. The terminal type parameter (TTP) controls the formatting of a message for a particular terminal type (for example, an IBM 2741 Communication Terminal). TTPPGBUF contains the address of a completed page.
5. The terminal output macro (DFHTOM) is issued to provide an open subroutine assembled within DFHTPP that puts a completed page out to the terminal. If the data stream contains extended attributes, and the terminal does not support extended attributes, the extended attributes are deleted.

DFHTPQ (undelivered messages cleanup program)

The undelivered messages cleanup program (DFHTPQ) checks the chain of automatic initiate descriptors (AIDs) to detect and delete AIDs that have been on the chain for an interval exceeding the purge delay time interval specified by the PRGDLAY system initialization parameter, if this has a nonzero value.

Figure 15 shows the undelivered messages cleanup program interfaces.

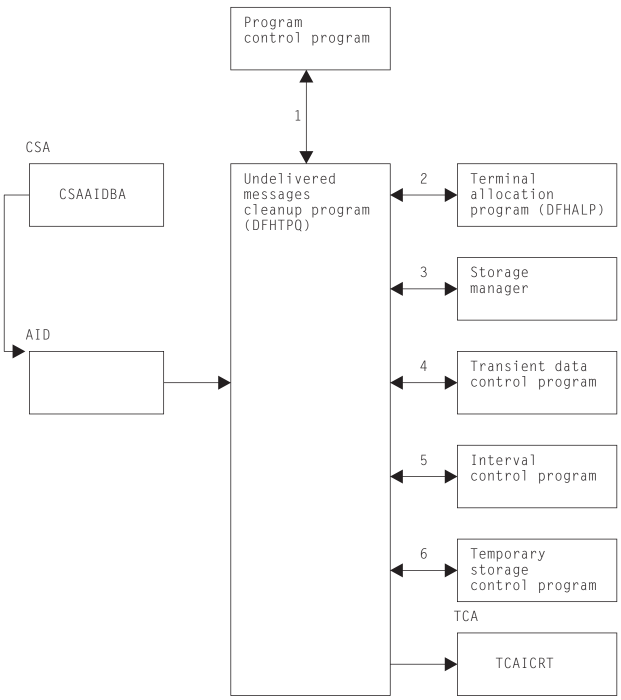


Figure 15. Undelivered messages cleanup program interfaces

Notes:

1. DFHTPQ is initiated the first time by the mapping control program (DFHMCP), by interval control, or by the transaction CSPQ. Thereafter, it reinitiates itself (see note 5).
2. DFHTPQ communicates with the allocation program (DFHALP) to locate and unchain AIDs.
3. DFHTPQ communicates with storage control to free AIDs that have been purged and to acquire storage for notification messages.
4. Transient data control is used to send notification messages.
5. Interval control is used to obtain the current time and to reinitiate this task (DFHTPQ).
6. DFHTPQ communicates with temporary-storage control to retrieve and replace message control records (MCRs) and to purge messages.

DFHTPR (terminal page retrieval program)

The terminal page retrieval program (DFHTPR) processes messages built by BMS and placed in temporary storage.

Figure 16 on page 49 shows the relationships between the components of page retrieval.

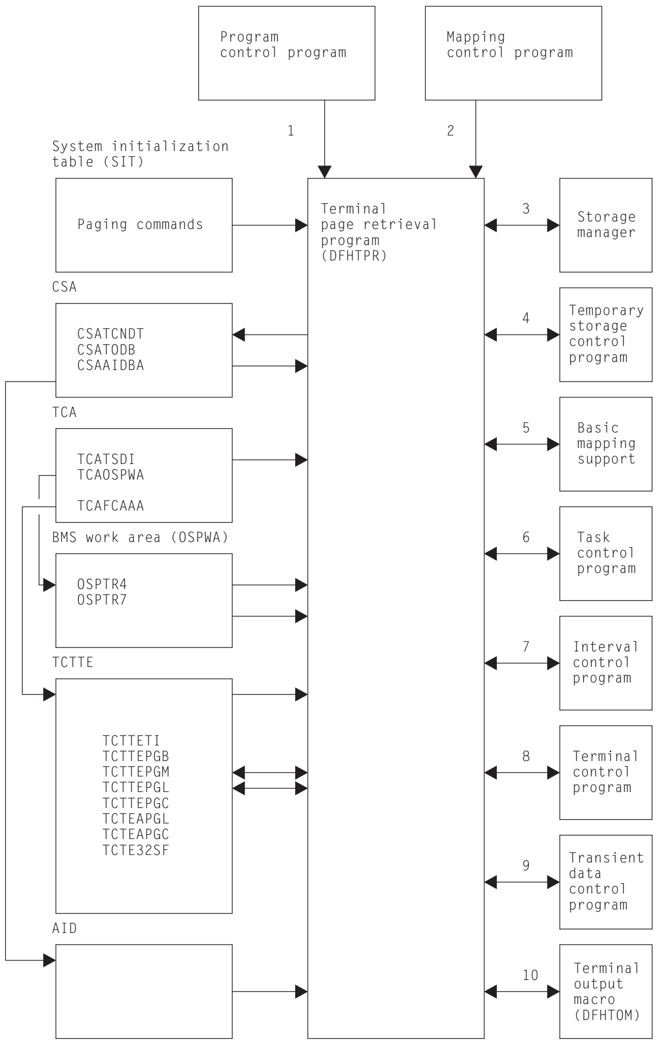


Figure 16. Page retrieval program interfaces

Notes:

1. DFHTPR can be initiated as a stand-alone transaction (CSPG), or by a user-defined paging command (for example, P/, or 3270 PA/PF keys), or linked to from a BMS conversational operation (SEND PAGE request with CTRL=RETAIN or RELEASE).

   DFHTPR performs the following functions:

   - Displays the first page of a routed message
   - Displays subsequent pages of a message at a terminal for which a SEND PAGE request with CTRL=AUTOPAGE was specified
   - Processes paging commands from a terminal
   - Processes the CSPG transaction when it is entered at the terminal
   - Purges a message displayed at the terminal if the terminal is in display status and other than a paging command is entered at the terminal.
2. DFHTPR is entered from the BMS mapping control program (DFHMCP) to display the first page of a message originated at the terminal if CTRL=RETAIN was specified in the BMS request. DFHTPR reads from the terminal and processes paging commands until other than a paging command is entered.
3. DFHTPR uses storage control to:

- Acquire and free message control blocks (MCBs)
- Free message control record (MCR) storage
- Acquire storage for information and error messages to be sent to the destination terminal and the master terminal
- Free an automatic initiate descriptor (AID) taken off the AID chain
- Acquire and free storage for a route list constructed in response to a COPY command entered at a terminal
- Acquire a TIOA into which to place a device-independent page when performing the COPY function.

4. Temporary-storage control is used to retrieve and replace MCRs and to retrieve and purge pages.
5. Basic mapping support is used to display error and information messages at a requesting terminal, and to send a page to the destination terminal in the COPY function.
6. Task control is used to retain exclusive control of an MCR while it is being updated.
7. DFHTPR communicates with interval control during error processing when a temporary-storage identification error is returned while attempting to retrieve an MCR. Up to four retries (each consisting of a one-second wait followed by another attempt to read the MCR) are performed. (The error may be due to the fact that an MCR has been temporarily released because another task is updating it. If so, the situation may correct itself, and a retry is successful.)
8. Terminal control is used to read in the next portion of terminal input after a page or information message is sent to the terminal when a SEND PAGE request with CTRL=RETAIN was specified.
9. Transient data control is used to send error or information messages to the master terminal.
10. The terminal output macro (DFHTOM) is issued to provide an open subroutine that puts a completed page out to the terminal.

DFHTPS (terminal page scheduling program)

The terminal page scheduling program (DFHTPS) is invoked for each terminal type to which a BMS logical message built with SEND MAP PAGING or SEND TEXT PAGING is to be sent. For each terminal designated by the originating application program, DFHTPR is scheduled to display the first page of the logical message if the terminal is in paging status, or the complete message if it is in autopage status.

Copy books

Copy book	Function
DFHBMSCA	Defines constants for field attribute values, flags returned by BMS, and character attribute types and values for SEND TEXT. It is usually copied into BMS application programs.
DFHMCPE	Included in the minimum-function BMS mapping control program DFHMCPE$, and also forms the BMS fast-path module DFHMCX used by both standard and full-function BMS. It is a small, fast, self-contained, limited-function BMS for 3270 displays and printers.
DFHMCPIN	Included in the standard and full-function versions of the BMS mapping control program, DFHMCPA$ and DFHMCP1$ respectively. It contains the code for input mapping.
DFHMIN	Included in the DFHM32 and DFHMCPE programs. It contains input mapping code for 3270 terminals.
DFHMSRCA	Defines constants for MSR control. This is usually copied into BMS application programs.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for basic mapping support, all with a trace level of BM 1:
- AP 00CD, for temporary-storage errors
- AP 00CF, for exit trace
- AP 00FA, for entry trace.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 6. Builders

The builder modules:
- Make the autoinstall process possible (that is, build a terminal control table terminal entry (TCTTE) dynamically).
- Allows new TCT entries to be added on a running CICS system.
- Allow the TCT to be dynamically updated on a running CICS system.
- Allow TCT entries to be deleted on a running CICS system.
- Reduce emergency restart times for those systems that use the autoinstall function. These systems have to take the time to restore and recover only those terminals that were autoinstalled at the time of termination.
- Reduce warm start times for those systems that use auto-install. No auto-installed terminals (except LU6.2 parallel systems are recovered at warm start).
- Reduce shutdown times for those systems using auto-install. Auto-install catalog entries are deleted but the entry in storage is not destroyed during shutdown.

In this section, the term TCTTE is used in a general way to refer to the terminal control table entries for connections (TCT system entries, TCTSEs), mode groups (TCT modegroup entries, TCTMEs), sessions (session TCT terminal entries, TCTTEs), skeletons (TCTSKs), and models.

To build or delete a control block for a particular device, a set of builders is called. The set of builders is specified by a tree structure of patterns, each pattern specifying one builder.

The builder modules (DFHBS*) are link-edited together into the DFHZCQ load module.

On microfiche, the individual DFHBS* modules are listed separately.

Design overview

What is a builder (DFHBS*)?

A builder is responsible for all the actions that can occur on a particular subcomponent of the TCTTE. The term subcomponent means a separately obtained area of storage which is referenced from the TCTTE or a collection of fields in the TCTTE that are logically associated with one another. General terms sometimes used instead of subcomponent are **object** or **node**. For example, the NIB descriptor, LUC extension, and BMS extension are all considered to be subcomponents.

Builder parameter set (BPS)

Each time a calling module invokes DFHZCQ for INSTALL, it supplies a builder parameter set (BPS). The BPS describes the device to be defined. The device-type is determined by matching attributes in the BPS with a table of definitions, DFHTRZYT, in module DFHTRZYP.

A BPS consists of a fixed-length prefix, a bit map preceded by its own length, an area for fixed-length parameters preceded by its own length, and three variable-length parameters, BIND, USERID, and PASSWORD. Each variable-length parameter has a 1-byte length field.

TCTTE creation and deletion

This section starts by describing the structure of the main components involved in the process of creating and deleting TCTTEs. Figure 17 on page 54 is in two halves: the top half shows those components that can initiate the process of collecting all the necessary data or parameters that go toward fully defining a TCTTE, and the bottom half is concerned with how to go about creating the TCTTE after it has the full set of parameters. Thus, all the processes are aiming for the same common interface. This section deals first with the top-level processes that are activated to create or delete TCTTEs; for the time being, assume that after returning from the DFHZCQ interface a TCTTE has been created. (For a more detailed description, see "DFHZCQ and TCTTE generation" on page 55.)

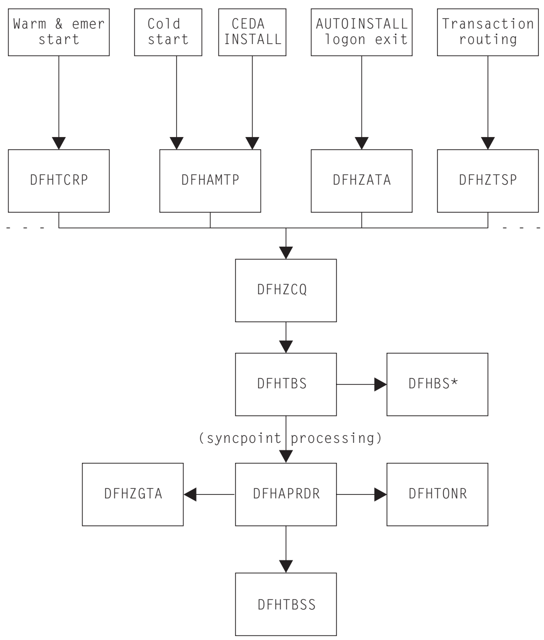


Figure 17. Top-level view of the components participating in TCTTE creation

Component overview

DFHTCRP

The DFHTCRP program is responsible for reestablishing the TCTTEs that were in existence in the previous run. There are conceptually three stages of processing in this module:

1. Initialize DFHZCQ. Initialize DFHAPRD. If START=COLD, terminate.
2. Reestablish TCTTEs that were saved on the CICS catalog. If START=WARM, terminate.
3. Call DFHAPRDR to forward-recover in-flight TCTTEs from the system log, if an emergency restart is being performed.

DFHAMTP

The DFHAMTP program is used as part of INSTALL processing. It calls DFHTOR, then DFHZCQ.

DFHZATA and the CATA transaction

CATA is a transaction that is initiated by the logon exit and causes DFHZATA to run. It is passed the CINIT which is used to deduce the parameters which must be passed to DFHZCQ in order to create a TCTTE.

DFHZTSP

The terminal sharing program, DFHZTSP, is used by transaction routing for devices of all types, exclusively so for non-APPC devices.

DFHZCQ

The DFHZCQ program supports the INSTALL and DELETE interface that results in the TCTTE being created or deleted. It relies on its callers to supply the complete set of parameters that are to be used to create the TCTTE; that is, it is not responsible for determining parameters for the TCTTE.

DFHBS* builder programs

The builders are responsible for creating the TCTTE. The parameters given to DFHZCQ are passed on to the builders. They extract the parameters and set the relevant fields in the TCTTE.

DFHTBS

The DFHTBS program is an interpreter that uses a pattern given to it by DFHZCQ to drive the whole TCTTE creation or deletion process according to certain rules.

DFHAPRDR

The DFHAPRDR program is the orchestrator of the commitment of TCTTE creation or deletion. It is responsible for driving DFHTBSS and DFHTONR for syncpoints, during cold start and also for recovering in-flight creates or deletes from the system log during emergency restart. It is called by the Recovery Manager, DFHTCRP and DFHAMTP during start-up and directly from DFHTBS (to roll-back an atom).

DFHTBSS

The DFHTBSS program is responsible for logging forward recovery records and for updating the catalog as a result of the request initiated by DFHZCQ and actioned by DFHTBS. It is driven by DFHAPRDR.

DFHTONR

The DFHTONR program is responsible for logging forward recovery records and for updating the catalog for install or delete requests for TYPETERMS. It is driven by DFHAPRDR.

DFHZGTA

DFHZGTA is the module called by DFHBS* and DFHZTSP (for remote system entry sessions) to add or delete index entries for TCTTE entries. It maintains locks on terminal namespaces, and handles calls to TMP to add, quiesce, delete, unlock and unquiesce entries. It is driven at syncpoint or rollback for an atom by DFHAPRDR.

DFHZCQ and TCTTE generation

This section describes how a TCTTE gets built and deleted. You need to understand at least one method by which a builder parameter set (BPS) is created; for example, CEDA INSTALL or AUTOINSTALL. A BPS contains all the values necessary for the creation of a TCTTE.

Figure 18 gives a more detailed view of the main components involved in the INSTALL process.

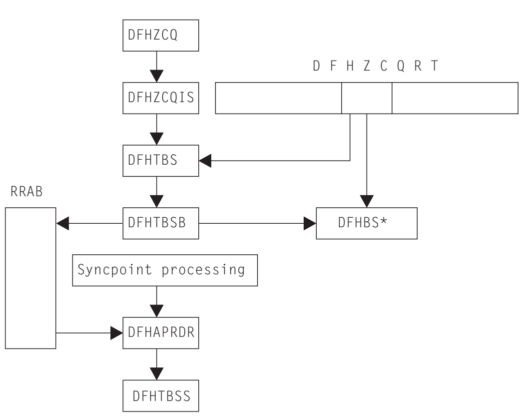


Figure 18. Major active components in the INSTALL process

The four-stage process

In summary, the process consists of four stages:

1. **Collecting the parameters** together.
2. **Creating the storage** for the TCTTE and copying the parameters. Note however, that at the end of this stage, a TCTTE has effectively been built. It is still unknown to the rest of the CICS system, that is, the TCTTE name has not been exposed. The modules involved here are DFHTBSB and DFHBS*.
3. **Producing a recovery record**. This is done at syncpoint processing time in the DFHTBSS module. This stage is usually called Phase 1 syncpoint.
4. **Writing or updating the catalog**. Again, this is done in DFHTBSS and is called Phase 2 syncpoint. It is at about this stage that the TCTTE name becomes exposed and known to the rest of CICS.

What is DFHZCQRT?

DFHZCQRT is an array of “patterns” where each pattern defines a list of builders that need to be called in order to create this particular type of TCTTE, that is, a pattern is equivalent to a type of terminal. The array entry consists of two parts: information that is private to DFHZCQ, and the pattern that is interpreted by DFHTBS.

What does DFHTBSBP do?

The pattern entry is passed to DFHTBSBP (via DFHTBSB) after it has been found by DFHZCQIS. DFHTBSBP calls each builder identified by the pattern in sequence to create the object for which the builder is responsible. Note that DFHTBSBP knows nothing about the TCTTE; DFHTBSBP merely follows a set of simple rules. It keeps an audit trail of each builder that is called.

What is the RRAB used for?

The audit trail kept by DFHTBSBP is implemented by obtaining a Resource definition Recovery Anchor Block (RRAB) that has some user storage attached to it. As DFHTBSBP calls each builder to perform an action, it adds an “action element” to the RRAB. (See “What is syncpointing?” on page 57) The address of the RRAB for a UOW is held in the ‘APRD’ recovery manager slot, which ensures that DFHAPRDR will be called at syncpoint. The RRAB stores the action blocks in two types of chains, one for actions that are not part of a named resource definition 'atom' and one for actions that are part of a named atom. This later type are chained off a Resource definition Action Name block (RABN). Also held in the RRAB is an indicator set by DFHTOR if DFHTONR should be called at syncpoint (if a typeterm has been installed), and a chain of Resource Definition Update Blocks (RDUB).

What is a resource definition 'atom'?

Certain resource definitions must be installed or deleted as a single set. These definitions are called a resource definition 'atom'. CICS installs the members of a RDO group as individual resource definitions, which can fail without causing the other resources to fail except for these atoms, which bear the name of the logical set of definitions. For example:

A connection and its associated sessions
is named for the connection

A pool of terminals
is named for the pool of terminals

What is a Resource definition Atom Name block (RABN)?

The RABN is only created for those atoms of resource recovery that are named. It holds the name of the atom, a chain of action elements for the atom, and the recovery outcome of the atom (whether it failed and was backed out, or succeeded and should be committed). DFHTBSB uses the RABN to decide if a session definition should not be installed because the install of the parent connection has already failed, for example. In our auto-install example, if the definition being installed is a parallel connection, there will be a RABN for it from which the action elements are chained.

What is a Resource Definition Update Block (RDUB)?

The RDUB is a record of locks held by a UOW against names in three namespaces:
1. Termids and Sysids
2. Netnames
3. Unique ids (Composed of the Netname of a Terminal Owning Region followed by a period '.' followed by the Termid or Sysid in that TOR)

During the installation, deletion, or replacement of a TCTTE definition the builders DFHBS* obtain locks by calling DFHZGTA. These locks guarantee exclusive or shared access to names in these namespaces. Exclusive access is used to prevent another task from installing another definition with the same name, netname or unique-id while this UOW is trying to install or delete (an action which may have to be reversed). Shared access is used to block another task from deleting an entry that a definition that this task is updating (for example, a system definition name may be locked by a remote terminal definition that refers to it).

RDUBs also exist on a global chain so that other UOWs can easily find out if a particular lock is held.

What is syncpointing?

When DFHTBSBP has exhausted the list of builders, it returns to its caller. Similarly, DFHZCQIS returns to its caller, which could have been autoinstall. However, there is still an audit trail that is attached to the RRAB. It is only when the calling task terminates or issues DFHSP USER or EXEC CICS SYNCPOINT that the next two stages occur.

Syncpoint processing consists of two phases. The first phase (prepare phase) requires the resource manager to write a forward-recovery record to the log. Thus, if the second phase (commit phase) fails to write to the catalog, this recovery record can be used to forward-recover the action on an emergency restart.

DFHTBS

The DFHTBS program is an interpreter that uses a pattern given to it by DFHZCQ to drive the whole TCTTE installation or deletion process according to certain rules.

DFHAPRDR

DFHAPRDR is invoked by recovery manager if the 'APRD' RM slot is non-zero. This slot contains the address of the RRAB for this UOW if any resource definition has taken place. It is also called by DFHTBS directly if an atom needs to be rolled-back or to commit an atom during Cold Start. DFHAPRDR examines the RRAB and chooses whether to call DFHTBSS, DFHTONR and DFHZGTA for each phase of syncpoint or individual atom commitment.

If either DFHTBSS or DFHTONR have records to log/catalog, DFHAPRDR calls the recovery manager to request that a record is written to the catalog noting that a forget record will be written once syncpoint completes. The purpose of this call is that if CICS should fail between the start of syncpoint phase 2 and the end, on an emergency restart recovery manager will call DFHAPRDR with the log records for this UOW so that they can be re-applied to the catalog, and the TCTTE entry or entries can be re-built.

DFHTBSS

The DFHTBSS program is responsible for performing the correct recovery actions for each atom and UOW at syncpoint (or during the rollback of an individual atom). It writes forward recovery records to the system log and updates the catalog during phase 1 and phase 2 of syncpoint respectively. It is directly driven by DFHAPRDR.

The purpose of the builder (DFHBS*) modules is to build a TCTTE, TCTSE, and TCTME and its associated control blocks. A TCTTE is built for terminals only; a TCTSE and TCTME are built for both LU6.1 with MRO and LU6.2 single sessions; all three are built for LU6.2 parallel sessions. DFHTBSS is invoked by DFHAPRDR with a parameter list that indicates whether this call is for an individual atom or for syncpoint and which phase is in force. For phase 1, it uses the action blocks audit-trail to recall each builder. It asks each builder to supply the address and length of the subcomponent so that it can create a single record containing a copy of each component as a list; that is, the first part of the record contains a copy of the object created by the first builder in the sequence, the second part contains a copy of the object created by the second builder, and so on until the audit trail list is finished. This record is then written to the system log as a forward recovery record.

When DFHTBSS is reentered for the second phase (again a parameter on the call by DFHAPRDR), it uses the record created in the first phase as the record that is written to the catalog. During this stage, each builder is called to tidy up after the object for which it is responsible; for example, for the TCTTE itself, it puts the TCTTE in service.

Again note, DFHTBSS only implements a set of rules.

DFHTONR

DFHTONR is responsible for writing catalog records for TYPETERMs. It is called by DFHAPRDR.

DFHZGTA

DFHZGTA is the module that is called by DFHBS* modules to add index entries for TCTTE entries so that they can be located quickly either by DFHZLOC, DFHZGTI or in VTAM exit code. It calls DFHTMP services. It obtains and releases locks using the RDUB blocks, and at syncpoint is responsible for releasing all TMP locks and unquiescing any TMP entries that were quiesced by DFHBS* modules.

Summary

- In overview, the process consists of four stages: parameter collection, obtaining and initializing, phase 1 recovery record and logging, and phase 2 catalog record.
- A builder contains TCTTE specific code.
- DFHTBS* modules implement the abstract rules for creating generic “objects”.
- DFHZCQRT contains patterns that define what builders are to be used to build the TCTTE.
- Syncpoint processing consists of two stages (prepare and commit).
- DFHAPRDR is responsible for orchestrating the syncpoint process for all of resource definition recovery.
- DFHTBSS is driven by DFHAPRDR using the audit trail produced by DFHTBSB.
- DFHTONR is driven by DFHAPRDR if any TYPETERMs were installed.
- DFHZGTA is driven by DFHAPRDR if any locks need to be released.

Example of an autoinstall

Consider the following: a terminal operator has logged on to the system and is being autoinstalled. The CATA transaction is responsible for collecting together the parameters required for the DFHZCQ INSTALL.

The process continues from the point where the DFHZCQ INSTALL is issued from CATA:

1. A call has been made to cause an install to occur. DFHZCQ ensures that other related modules are already loaded.
2. DFHZCQ calls the install-specific module (given in the parameter block passed to DFHZCQ)
3. DFHZCQIS performs various checks on the parameters passed by the caller of DFHZCQ.
4. DFHZCQIS finds a pattern in DFHZCQRT that matches with information given in the parameters.
5. DFHZCQIS calls DFHTBS with the pattern and parameters.
6. DFHTBS routes the request to DFHTBSB; it is omitted from further discussions.
7. DFHTBSB checks that a valid pattern has been passed.
8. DFHTBSB creates the RRAB which gets attached to the APRD Recovery Manager slot.
9. DFHTBSB calls the next builder as defined by the pattern.
10. Each builder (DFHBS*) creates its section of the TCTTE.

11. DFHTBSB adds an action element to the RRAB giving information about this particular builder.
12. Steps 9 on page 58, 10 on page 58, and 11 are repeated until the pattern is finished.
13. DFHTBSB tidies up the RRAB and returns.
14. DFHTBS returns.
15. If the return code was 'OK', DFHZCQIS returns the address of the hidden TCTTE.
16. DFHZCQ returns.
17. The caller continues until DFHSP USER is issued or the task terminates.
18. DFHAPRDR invokes DFHTBSS with the RRAB indicating phase 1.
19. DFHTBSS examines the RRAB to determine phase.
20. Using the action elements created in step 11, DFHTBSS recalls each builder asking for information to be saved on the recovery log.
21. Each builder (DFHBS*) returns the address of the object built in step 10 on page 58.
22. Using these addresses, DFHTBSS builds the recovery record.
23. DFHTBSS writes the recovery record to the system log.
24. DFHTBSS saves the stored version for the next phase.
25. DFHTBSS returns.
26. Recovery Manager calls all other resource managers that have a part to play in the process; it knows this because there are addresses in the RM slots for this UOW.
27. DFHTBSS is called for phase 2. It reuses the in-storage version of the recovery record to write to the catalog.
28. DFHTBSS returns.

Patterns, hierarchies, nodes, and builders

Patterns were introduced in the previous section. This section examines in detail what they look like. To achieve this, several terms have to be explained.

What is a hierarchy?

In this context, “hierarchy” is another word for tree. The structure of the TCTTE can be thought of as a tree: at the top **node** is the TCTTE itself, containing pointers to lower-level **nodes**.

Figure 19 shows the **master node** as the TCTTE, with **subnodes** connected to it (BMS extension, special features extension, and so on).

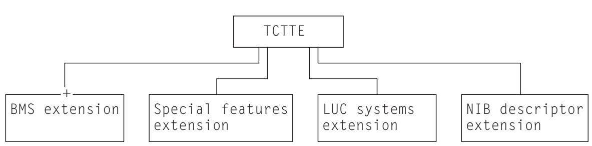


Figure 19. TCTTE structure

As a result of this structure, it can be seen that the creation process must follow several rules. For example, the storage for the **master node** has to be obtained before pointers to **subnodes** are saved in it.

What is a pattern?

The objective of a pattern is to reflect or represent the hierarchy as described above. Figure 20 outlines the shape of a pattern. For each of the nodes in Figure 19, there is a pattern. Starting with the TCTTE (**the master node**), there is a **master pattern**. B1offset references the **subpattern** for the BIND image node; B2offset references the subpattern for the BMS extension node; B3offset and B4offset reference the subpatterns for user area and SNTTE **subnodes** respectively. In total, there are five patterns: the master pattern and four subpatterns—so what is meant by **pattern** above was really a collection of patterns.

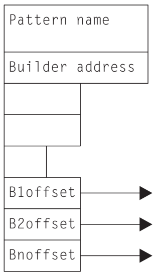


Figure 20. Pattern structure

Note that each pattern contains the address of a builder, so we could represent the TCTTE structure as:

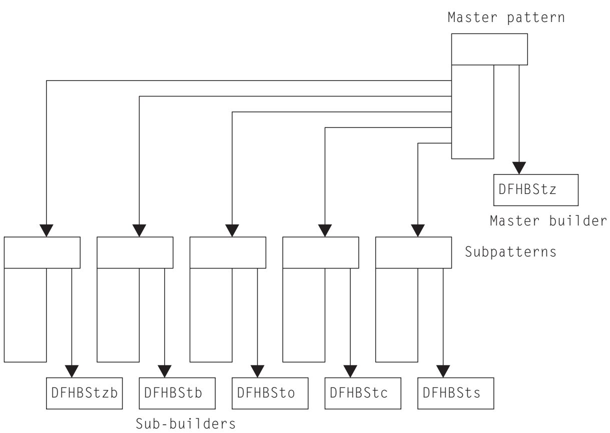


Figure 21. Patterns and subpatterns

The purpose of the builders

The purpose of the builders is to centralize the major functional code for creation and deletion of the **nodes** associated with the TCTTE. Figure 20 and Figure 21 show how the **patterns** refer to the builders; the pattern is exploited by the DFHTBS* code to activate the relevant builder function. For example, DFHTBSBP, when given a pattern, extracts the address of the builder and invokes the BUILD function belonging to the builder.

How does DFHTBSBP do its work?

First, you must examine more closely the structure of a builder in Figure 22 on page 61.

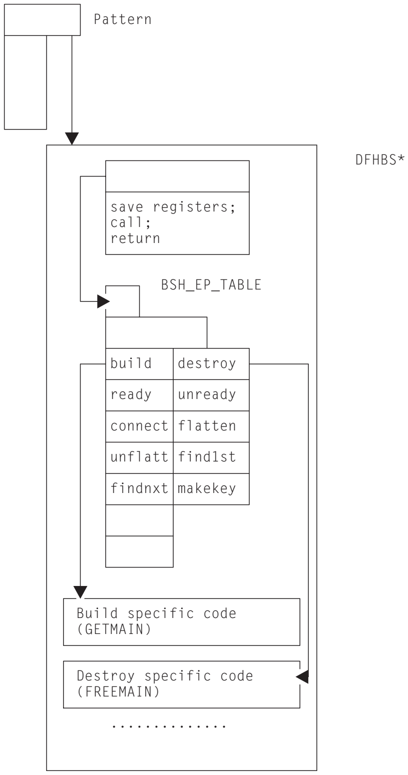


Figure 22. The builder stub

Remember that the pattern references a builder. In fact, it references a stub, the first word of which points to a table (BSH_EP_TABLE), and is followed by code that is responsible for enacting the entry as required by the caller. For example, if the caller wanted to call BUILD, a call would be made to the stub with value 1. The stub would extract the offset to the build code from the BSH_EP_TABLE, and perform the call.

Thus, making a call from DFHTBS* to DFHBS* is relatively simple: all that is needed is the function number (1 for BUILD, 2 for DESTROY, ...), a call to the stub, and the pattern.

Summary

- The TCTTE is structured as a **hierarchy** with a **master node** (the TCTTE itself) and **subnodes** (BIND image, BMS extension, and so on).
- **Patterns** mimic this hierarchy and consist of a **master pattern** which refers to **subpatterns**.
- In turn, each pattern points to a builder: the master pattern refers to the **master builder** and the subpatterns refer to the **sub-builders**.
- Builders centralize the major creation and deletion functions associated with the node for which they are responsible.
- The invocation (or activation) of the builder functions is performed under the strict control of the DFHTBS* modules.
- The **order of invocation** is totally determined by the structuring of the patterns.

The DELETE process

By examining the hierarchy (see Figure 19 on page 59), you can see that there are certain rules that have to be established. Firstly, you should check that the TCTTE and its subcomponents are quiesced, that is, there is no activity in progress. And secondly, and perhaps more obviously, the top node must not be the first object to be freed. From this, you can derive two basic rules, or “functions”, that must be supplied by any DFHBS*:

UNREADY
For all nodes associated with the master node. Ensures that no activity is occurring; for example, that a CLSDST is not in progress. It must also achieve exclusive ownership of the object; for example, ZGTA QUIESCE ensures no locates on the given TCTTE succeed and that no other UOWs can install another similarly named object until syncpoint. Further, it **initiates** the ZGTA DELETE which does a TMP DELETE to remove the entry.

DESTROY
Lower objects first. (See “What about the “lower objects first” rule?”:) Frees the storage belonging to the node.

What about the “lower objects first” rule?

Figure 23 tries to add meaning to the descriptions of the UNREADY and DESTROY functions. As each builder is called (as determined by the master pattern), DFHTBSD records an audit trail of called builders. However, the audit trail is managed slightly differently for the delete process, to guarantee order of processing by DFHTBSS at phase 2 time. For further information, see “Completing the process description” on page 63.

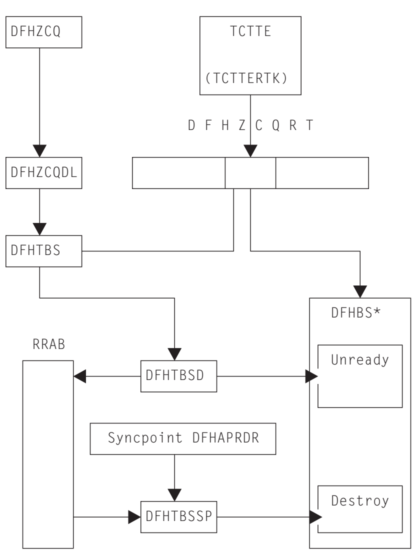


Figure 23. Major active components in the DELETE process

Example of a reinstall

1. CEDA reads the CSD and converts the definition into a builder parameter set (BPS).
2. CEDA issues a DFHZCP INSTALL passing the BPS.
3. Using the resource type code in the BPS, DFHZCQIS searches the DFHZCQRT table for the associated pattern.
4. DFHZCQIS calls DFHTBSB passing the BPS and the pattern.
5. DFHTBSB checks the pattern and creates a resource definition recovery action block (RRAB) for the audit trail.
6. Using the pattern, DFHTBSB calls the CHECKSET entry point of the associated builder.
7. The master builder does a DFHZGTI LOCATE to check whether the TCTTE already exists.
8. A TCTTE is found to exist, so the builder issues DFHZCP DELETE passing the address of the old TCTTE.

9. When a TCTTE is created, its position within the DFHZCQRT table is saved in the TCTTE. DFHZCQDL uses this value to find the pattern associated with this TCTTE.
10. DFHZCQDL calls DFHTBSD passing the object to be deleted and the pattern.
11. DFHTBSD extends the audit trail so that information about this delete can be recorded.
12. DFHTBSD calls the UNREADY entry of each builder.
13. Each builder (DFHBS*) checks whether its part of the TCTTE is being used (and vetoes the UNREADY if it is). It calls ZGTA QUIESCE and ZGTA DELETE to lock and remove the index entries.
14. DFHTBSD updates the audit trail for each called builder.
15. DFHTBSD returns.
16. DFHZCQDL returns.
17. The master builder checks the return code (that is, that no builder vetoed the UNREADY).
18. The master builder returns.
19. DFHTBSB checks the return code and recalls each builder at the BUILD entry point passing the BPS.
20. Each builder obtains some storage and copies the parameters from the BPS. It uses ZGTA ADD calls to lock and add index entries
21. DFHTBSB tidies up the RRAB and returns.
22. DFHZCQIS records the position within DFHZCQRT that enables DFHZCQDL to find the pattern.
23. DFHZCQIS Returns.
24. CEDA checks the return code and issues DFHSP USER.

Note: At this stage there are two TCTTEs: the old one that was UNREADY and the new one.

25. CEDA calls: DFHTBSS is entered for the first time (phase 1). The audit trail consists of two parts (A and B). Part A contains the list of builders involved with the UNREADY; part B contains the list of builders that created the new TCTTE.
26. CEDA writes a recovery record to the system log for Part A indicating that a delete is about to take place in phase 2.
27. CEDA creates a recovery record from Part B which represents the new TCTTE to be built.
28. CEDA calls each builder asking for its subcomponent (FLATTEN).
29. DFHZQIX returns an address and length.
30. CEDA concatenates each subcomponent into the recovery record.
31. CEDA writes the recovery record to the system log.
32. CEDA returns (end of phase 1).
33. CEDA reenter for phase-2 processing.
34. CEDA processes Part A, calling the DESTROY entry for each builder.
35. Each builder frees its part of the old TCTTE.
36. CEDA processes Part B of the audit trail.
37. CEDA writes the recovery record to the catalog.
38. CEDA calls the READY entry point for each builder on the audit trail.
39. Each builder does any tidying up that needs to be done.
40. CEDA returns.

Completing the process description

To complete the description of the creation and deletion process, two further functions must be described: CONNECT and READY.

CONNECT

Figure 19 on page 59 shows the TCTTE hierarchy. All that has happened at build time is that the separate parts of the TCTTE have been obtained. Access to these subcomponents is achieved by referencing pointers that are held in the TCTTE. So the CONNECT builder entry point is used to join the subcomponent to the TCTTE.

READY

The READY builder entry point is provided to enable any final tidying up that may be required at the end of the build process. For example, if the TCTTE has the AUTOCONNECT option, a SIMLOGON is initiated from this entry point. In general, this entry point is rarely used.

The creation/deletion state machine

Figure 24 shows the symmetry between the various builder functions.

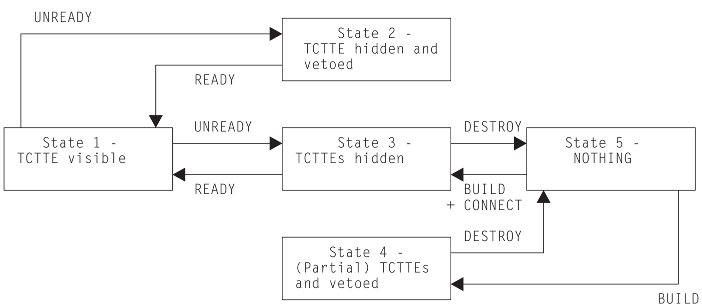


Figure 24. Create/delete state diagram

The starting point can be either state 5 (installing a TCTTE) or state 1 (deleting a TCTTE). Thus, if several TCTTEs had been successfully built, but the last one resulted in an error, we would end up in state 4. If it were not for the last one, we would have ended up in state 3. So the caller is returned an error response, and issues a DFHSP ROLLBACK. This causes DFHTBSS to call the DESTROY function of the builders for all elements on the audit trail—even for those that were “successfully” built in this atom, or UOW. Thus, an install of a atom can be perceived as one complete unit. During the DESTROY process, if the atom is being rolled-back, the builders call ZGTA QUIESCE and ZGTA DELETE to remove index entries for the new TCTTE. Likewise during the READY process, if a delete is being rolled back, the builders call ZGTA ADD to re-instate index entries for the TCTTE.

The hierarchy and its effect upon the creation process

Summary so far

- Object creation is a four-stage process.
- It is controlled by a pattern.
- Each pattern refers to a builder.
- Each builder is responsible for a subcomponent of the TCTTE.
- Builders have a number of procedural entry points:
  - BUILD
  - CONNECT
  - DESTROY
  - READY
  - UNREADY.
- These entry points are called under the control of the DFHTBS components.

This section now looks in greater detail at how the control of the builder calling process is implemented. To do that, you need to understand in greater detail the structure of the hierarchy, and the way the DFHTBS components interpret that structure.

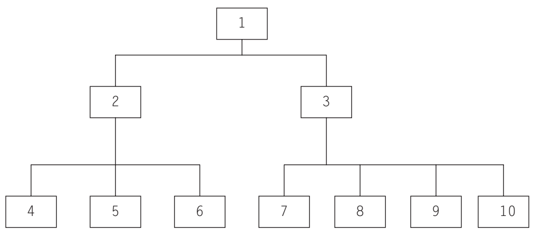


Figure 25. A general hierarchy

Figure 25 shows a more general hierarchy. Node 1 can be considered as a master node: it is at the top of the tree and has two subnodes (2 and 3). However, you could say that node 2 and its subnodes are also a tree: node 2 is the master node, and nodes 4, 5, and 6 are the subnodes. Similarly, with node 3: it has subnodes 7, 8, 9, and 10.

The DFHTBS components exploit the idea that a tree consists of a node with trees below it. In fact, DFHTBSBP uses **recursion** to access the tree of patterns.

Recursion

This section demonstrates how recursion is used to process a much simpler structure than that given in Figure 25. The example shown in Figure 26 on page 66 is for the DFHTBSP program, which has the following parameters:

Input: PATTERN, HIGHERNODE, and BUILDER
Inout: AUDITTRAIL
Output: NODE and RESPONSE.

The following list outlines the flow in DFHTBSBP. The step references refer to steps in this list.

1. Add and initialize an action to the AUDITTRAIL (this is used later in steps 5 and 11).
2. Using parameter PATTERN, find the address of the associated builder.
3. Call the builder stub with function number 1 (for BUILD) with the following parameters:
   Input: HIGHERNODE and BUILDER
   Output: NODE.

The builder uses the BUILDER parameters to create its specific object. Storage is obtained and the parameters are copied into it.

4. Check that the response from the build is ‘OK’.
5. Copy the address of the output parameter NODE into the AUDITTRAIL action.
6. Process all the subpatterns that may be attached to your pattern
7. Get the next subpattern Pn.
8. Call DFHTBSBP with the following parameters:
   Input: Pn, NODE, and BUILDER
   Inout: AUDITTRAIL
   Output: SUBNODE and SUBRESPONSE

   Note: In this step, you call yourself again, passing NODE. At the next level of recursion, this appears as HIGHERNODE.
9. Stop when the last pattern is processed.
10. Call the builder stub with function number 5 (for CONNECT) with the following parameters:
    Input parameters: NODE

Inout parameters: HIGHERNODE

The builder's CONNECT entry point now places the address as given by NODE into an offset of HIGHERNODE.

11. Finally, place the address of the pattern into the AUDITTRAIL action.

Simple recursion example

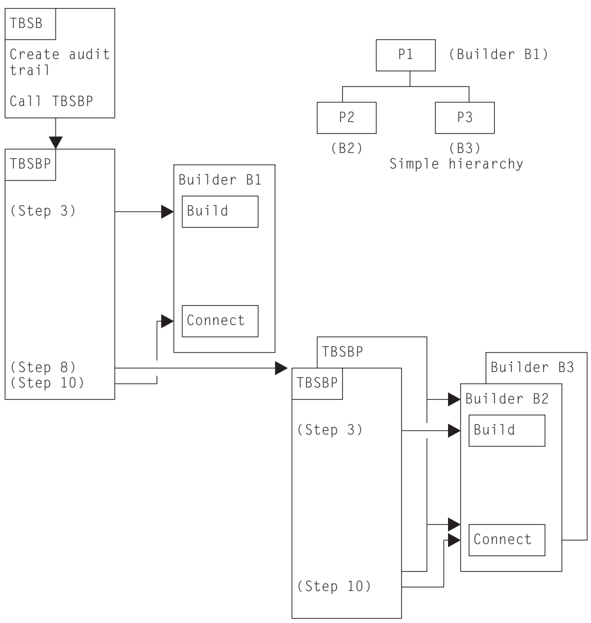


Figure 26. Simple example showing recursion

Consider the following simplified version of the hierarchy as given in Figure 26. The step references refer to steps in the list in the section "Recursion" on page 65.

1. Start with pattern P1. Call its associated builder (step 3 on page 65). This creates node N1.
2. All the patterns below P1 are processed, the first of which is P2.
3. Call DFHTBSBP passing P2, N1, BUILDER parameters, and others:
   a. Using the passed pattern (now P2), call the builder. This creates node N2.
   b. Process all patterns below P2; there are no subpatterns, so steps 6 through 9 on page 65 are not performed.
   c. Call the CONNECT entry of the builder, passing higher node N1 and the node just created, N2. This makes N1 point to N2.
   d. Return to caller.
4. Get the next pattern, P3.
5. Call DFHTBSBP passing P3, N1, BUILDER parameters, and others:
   a. Using the passed pattern (now P3), call the builder. This creates node N3.
   b. Process all patterns below P3; there are no subpatterns, so steps 6 through 9 on page 65 are not performed.
   c. Call the CONNECT entry of the builder passing in higher node N1 and the node just created N3. This makes N1 point to N3.
   d. Return to caller.
6. Last pattern processed (step 10 on page 65).

7. Call the builder associated with P1 to connect node N1 to HIGHERNODE. (This is zero because there is no higher node. Usually, a master builder's CONNECT function either does nothing or adds the TCTTE name and address into the table management tables.)

ROLLBACK

What happens when an error occurs during the install process? An example of this would be when one TCTTE within a group is still in service when a CEDA COPY command is being processed for the group with the REPLACE option specified. "Example of a reinstall" on page 62 shows such a replace operation. The builders for the existing TCTTE are called (UNREADY) in order to check that the DELETE (FREEMAIN) can proceed. Thus, the audit trail refers to all called builders.

If the "total vote" from all the UNREADY builder calls indicates OK, the build proceeds for the new TCTTE that is to replace the existing one. Thus, at the end of the process, the audit trail consists of a list of references to builders associated with the old TCTTE, and a list of references to builders for the new TCTTE (lists A and B).

Consider the case when the group contains definitions for three TCTTEs, and a VETO occurs for the last one. This means that there is an audit trail for A1, B1, A2, B2 for which there was success, and list A3 for the unsuccessful UNREADY for the third TCTTE.

The failure condition is returned to the caller (CEDA), which then issues a DFHSP ROLLBACK.

Recovery Manager invokes DFHAPRDR which in turn invokes the DFHTBSS module, with a parameter that indicates a rollback is required. Thus, the "A" lists are processed, and all the READY entry points of the builders are called. Then the "B" lists are processed, and the DESTROY builder entry is called to free the storage obtained for the supposedly new TCTTEs.

To summarize, the rollback operation for UNREADY is READY, and the one for BUILD is DESTROY.

Catalog records and the CICS global catalog data set

Overview

The fourth stage of the process is to produce a catalog record that is written to the CICS global catalog data set. This record is used on a subsequent restart to recreate the TCTTE, but in a different way from the "Build" process described above. A CEDA INSTALL means that the TCTTE lives across CICS restarts, avoiding the necessity of rerunning the install.

A RESTORE from the CICS catalog is a faster operation than a CEDA INSTALL because there is no conversion of the CSD definition to a builder parameter set, and less I/O involved.

In summary, a catalog record is produced by recalling each of the builders asking for the address of the data that they want to be recorded on the catalog. Each subcomponent of the TCTTE is then copied and concatenated into one record, which is then written to the catalog. This process is known as FLATTEN.

A CATALOG call is made when significant events change the state of a TCT entry which would be needed on a subsequent emergency restart. An example is the recording of the membername of a generic VTAM resource connection when a bind has occurred for the first time.

On the restart, the record is read from the catalog, and presented back to each of the original builders. Each builder performs a GETMAIN, and copies its section of the recovery record into the acquired storage. This process is known as UNFLATTEN.

At shutdown, auto-installed entries are removed from the catalog with an UNCATALOG call (if they were cataloged because AIRDELAY¬=0). This drives DFHTBS and the builders to produce similar records to those for a DELETE call, but only to take action to delete the catalog record. This is significantly more efficient than calling the builders to DELETE each entry, as the copy in storage is left untouched.

The key and the recovery record

When the build process in DFHTBSBP has finally finished, this module makes a call to the master builder at the MAKEKEY entry point. The builder produces a key that is used to index the associated recovery record. (See Figure 27.)

This information is placed on the audit trail so that it can be picked up by DFHTBSS. It consists of two parts:
1. Information that allows access to the catalog
2. The recovery record header.

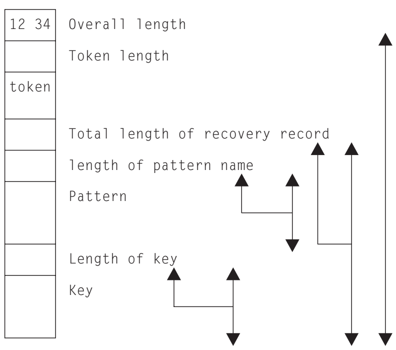


Figure 27. The recovery record

More about the audit trail

Figure 28 on page 69 shows the layout of an audit trail. Internally it is known as an **action block**, which consists of **action elements**. As each builder is invoked by DFHTBSBP or DFHTBSDP, an action element is appended to the action block. Each element has a reference to a pattern (PATT). This is to allow DFHTBSS to enter the associated builder at the READY or DESTROY entry points.

CCRECP contains the address of the recovery record header. Only one of these is produced as a direct result of the MAKEKEY call to the **master builder**. All other action elements have their CCRECP set to zero.

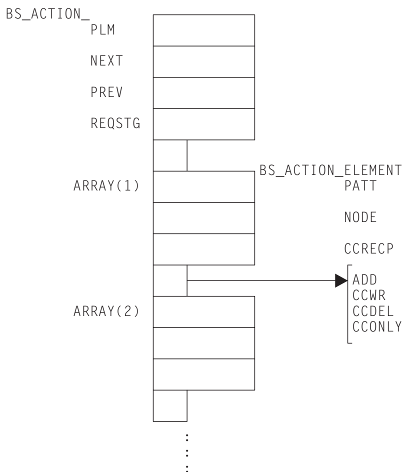


Figure 28. Action block and action elements (audit trail)

DFHTBSS and the FLATTEN process

During phase-1 syncpoint processing, DFHTBSS searches the action elements for a nonzero CCRECP. On detection, it calls DFHTBSLP, passing the reference to the pattern as given in the action element.

The storage “segments” are returned to DFHTBSSP which extracts the address and length from each segment and copies them into the recovery record.

The RESTORE process

The recovery record header contains the pattern name which is used to find the master pattern in DFHZCQRT. This is then passed to DFHTBSR to drive the recovery process by calling each builder’s UNFLATTEN entry.

Each segment is extracted from the recovery record and is passed to the associated builder’s UNFLATTEN entry point. These routines usually obtain some storage and copy the segment into it.

Control blocks

Builder modules all use both LIFO and a builder parameter set (BPS), which are passed between the CSECTs (DFHBS* modules). See “Builder parameter set (BPS)” on page 53 for further information about the BPS.

Terminal storage acquired by the builders

The following terminal storage is acquired by the builders:

Control block field	Description	Storage manager subpool
TCTSE	Terminal control table system entry	ZCTCSE
TCTME	Terminal control table mode entry	ZCTCME
TCTTE	Terminal control table terminal entry	ZCTCTTEL (large TCTTEs) ZCTCTTEM (medium TCTTEs) ZCTCTTES (small TCTTEs)
TCTENIBA	NIB descriptor	ZCNIBD
TCTEBIMG	BIND image	ZCBIMG
TCTTECIA	User area	ZCTCTUA
TCTTESNT	Signon extension	ZCSNEX

TCTELUCX	LUC extension	ZCLUCEXT
TCTTETEA	BMS extension	ZCBMSEXT
TCTTETPA	Partition extension	ZCTPEXT
TCTTECCE	Console control element	ZCCCE

TCTTE layout

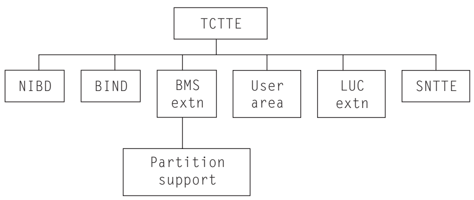


Figure 29. TCTTE layout

Formatted dumps give the TCTTE first, followed by its supporting control blocks.

Terminal definition

CEDA DEFINE puts a definition on the CSD. The definition is in the form of a CEDA command.

CEDA INSTALL reads the definition from the CSD, calls the builders and builds the definition in CICS DSA, and updates the CICS global catalog data set for future recovery.

EXEC CICS CREATE builds the same record that would be obtained from the CSD and then calls the builders just like CEDA INSTALL.

EXEC CICS DISCARD calls the builders with a pointer to the TCTTE entry that is to be deleted. The builders then freemain the TCTTE, remove index entries and the catalog record.

Modules

DFHZCQ handles all requests for the dynamic add and delete of terminal control resources. It contains the following CSECTs:

DFHBSIB3	DFHBSSZM	DFHBSTP3	DFHBSTZ1
DFHBSIZ1	DFHBSSZP	DFHBSTS	DFHBSTZ2
DFHBSIZ3	DFHBSSZR	DFHBSTT	DFHBSTZ3
DFHBSMIR	DFHBSSZS	DFHBSTZ	DFHBSXGS
DFHBSMPP	DFHBSSZ6	DFHBSTZA	DFHBSZZ
DFHBSM61	DFHBST	DFHBSTZB	DFHBSZZS
DFHBSM62	DFHBSTB	DFHBSTZC	DFHBSZZV
DFHBSS	DFHBSTBL	DFHBSTZE	DFHZCQCH
DFHBSSA	DFHBSTB3	DFHBSTZH	DFHZCQDL
DFHBSSF	DFHBSTC	DFHBSTZL	DFHZCQIN
DFHBSSS	DFHBSTD	DFHBSTZO	DFHZCQIQ
DFHBSSZ	DFHBSTE	DFHBSTZP	DFHZCQIS
DFHBSSZB	DFHBSTH	DFHBSTZR	DFHZCQIT
DFHBSSZG	DFHBSTI	DFHBSTZS	DFHZCQRS
DFHBSSZI	DFHBSTM	DFHBSTZV	DFHZCQRT
DFHBSSZL	DFHBSTO	DFHBSTZZ	DFHZCQ00

Note: The term “node” refers either to a TCTTE or to one of its subsidiary parts, such as the NIB descriptor.

Subroutines that are found in the builders:

BSEBUILD
BUILD: Create the node. For example, obtain the shared storage for the node.

BSECON	CONNECT: Connect the higher node to the lower. For example, make the TCTTE point to the NIB descriptor.
BSEDESTR	DESTROY: Abolish a deleted node. For example, free the storage removed from TMP's chains.
BSEFINDF	FINDFIRST: Find the first subsidiary node of a higher node. For example, BSFINDF(TCTTE) returns the NIBD being built.
BSEFINDN	FINDNEXT: Find the next subsidiary node of the one just found. For example, return the address of the next model TCTTE.
BSEFLAT	FLATTEN: Build the catalog or log record segment for each part of the TCTTE. This is passed back to the caller to create a complete "flattened" TCTTE.
BSEMAKEY	MAKEKEY: Create a key that is used to write out the new node to the global catalog.
BSENQIRE	ENQUIRE: The converse of BUILD, it creates a BPS from a TCTTE. The BPS can then be shipped to another system.
BSEREADY	READY: Make a node ready to use. For example, add to TMP's chains.
BSERESET	RESET: Build the TCTTE from the CICS global catalog. (RESET is a cut-down version of UNFLATTEN.)
BSEUNFLA	UNFLATTEN: Build the TCTTE from the CICS global catalog.
BSEUNRDY	UNREADY: Check that a node can be deleted. For example, ensure that no AIDs are queued on a TCTTE before deleting.

Not all subroutines are found in all builders. Certain subroutines are required, but do nothing other than return to the caller. The subroutine names are the same in each builder.

Module entry

Consider a module entry to be a router that does some housekeeping and then branches to the appropriate subroutine:

- Enter the builder at offset X'18'.
- The first X'17' bytes are taken up by the standard DFHVM macro expansion.
- Save DFHTBS's registers (DFHTBS calls each builder).
- Save the first two entries in the parameter list:
  1. The address of LIFO storage
  2. The index number of the subroutine to call.
- Increase the value of register 1 by 8 to get past the first two entries.
- Branch to the appropriate subroutine of the builder using the index number passed.
- Return from the builder subroutine.
- Restore registers.
- Return to DFHTBS.

Subroutine entry

- Register 1 points to the parameter list.
- Store Register 14 (return address) at Register 2 + X'nn' (varies by entry point).
- Store the parameter list into Register 2 + X'nn' (varies by entry point).
- The length of the parameter list varies.

Subroutine exit (return to module entry)

- Exit from the subroutine only through an "official" exit point.
- The exit point is usually the end of the subroutine.
- The end of the subroutine is indicated with "*end; /*BUILD */".

- In some cases, the end of the subroutine branches back to the exit point somewhere within the subroutine.
- Return (BR R14) from within the subroutine.
- Reload Register 14 from Register 2 + X'nn' and return to caller.

Patterns

In DFHZCQRT, a series of patterns define the flow through the builder modules. (See Figure 30.) For each kind of terminal, there is a different pattern.

If installing, DFHZCQIS selects the pattern and calls DFHTBS (table builder service). If deleting, DFHZCQDL does the selection.

DFHTBS interprets the pattern and calls each builder that the pattern calls out. DFHTBS knows nothing about the terminal or whether you are installing or deleting. It simply does what the pattern tells it to do.

DFHTBS passes a BPS as it calls each builder. The BPS allows one builder to be used for many different kinds of terminals. For example, DFHBSTC obtains the user area for all terminal types. The BPS contains the length to be obtained.

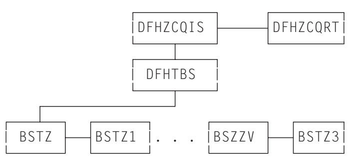


Figure 30. Calling sequence of builders (determined by patterns)

Calling sequence of builders for a 3277 remote terminal

1. DFHZCQRT contains a series of comments followed by the patterns. The comment appears as:

   ```
   /* * * * * * * * * * * * * */
   /*      3277 REMOTE      */
   /* * * * * * * * * * * * * */
   ```

2. Shortly afterwards is a Declare (DCL) followed by a level-1 name:

   ```
   DCL  1 P145002  STATIC
   ```

   This is the name of the pattern that drives the build process for a 3277 remote terminal.

   - DFHBSTZ is indicated to be the first builder called.
   - One pattern is used to drive the building process.
   - 18 subpatterns are to be used.
   - Three of these 18 subpatterns each call one additional pattern.
   - The terms “pattern” and “builder” mean the same thing. Therefore:

     ```
     DFHBSTZ  +  DFHBSxx  +  DFHBSxx  =  22
       (1)    +    (18)   +    (3)    =  22
      pattern +    sub-   +  sub-sub- =  22
                 patterns    patterns
     ```

     Thus we have to go through 22 builder modules to build a 3277 remote terminal.

3. Go to the cross-reference at the back of the dump and find where P145002 is defined in assembler language. Go to that address.
4. This states that the first builder to be called is DFHBSTZ. This is the main one.
5. Drop down to the 2-byte fields that follow: these state the names of the builders that are to be called, in sequence (18 should be listed).
6. The first one is IAATZ1 which does not sound familiar:

- Go to the cross-reference at the back of the dump, look up IAATZ1, and go to where it is defined.
- You see that this is DFHBSTZ1.
- You can also see a close resemblance between IAATZ1 and DFHBSTZ1, but do not count on this to be always true.

7. Now you know that the second builder to be called is DFHBSTZ1.
8. The next two builders to be called are IAATCV (DFHBSTV) and IAATCB (DFHBSTB).
9. The fifth builder to be called according to the pattern needs to be looked at:
   - The pattern says that IACTZ3 should be called.
   - When you go to where IACTZ3 is defined, you find that this is DFHBSIZ3.
   - You also see that DFHBSIZ3 calls IAATM.
   - Look up IAATM and you see that it is DFHBSTM.
   - This is a sub to a subpattern, and this is how nesting of builder calls occurs.
   - Thus, DFHBSIZ3 calls DFHBSTM when building a local 3277.
   - DFHBSTM accounts for one of the "other" three mentioned in step 2.
10. If you continue through this pattern, you can identify the names of the 22 builders that would be called to build a 3270 local TCTTE.

    Here is the complete list, in order, of the builders that are called:

```
1  DFHBSTZ         12 DFHBSTH
2  DFHBSTZ1        13 DFHBSTI
3  DFHBSTZV        14 DFHBSTS
4  DFHBSTZB        15 DFHBSTT
5  DFHBSIZ3        16 DFHBSTZA
6  DFHBSTM         17 DFHBSTP3
7  DFHBSTB         18 DFHBSZZ
8  DFHBSIB3        19 DFHBSTB3
9  DFHBSTO         20 DFHBSTZE
10 DFHBSTC         21 DFHBSZZV
11 DFHBSTE         22 DFHBSTZ3
```

    A look at "Pattern Trace" supports this flow. Note that the first ZCP TBSB(P) BUILD and its matching return (the return has no builder suffix) should be ignored.

Builder parameter list

As each builder is called by DFHTBS, a parameter list is passed. Unique data is passed to enable one builder module to be called for a variety of terminal types. The length of the builder parameter list is fixed for each kind of subroutine; for example, the parameter list passed to BSEBUILD is always X'23' bytes long, regardless of the builder involved.

```
Subroutine     Length of parameter list
               (hexadecimal)
BSEBUILD       23
BSECON         13
BSEDESTR        7
BSEMAKEY        B
BSEREADY        3
BSEUNRDY       17
BSEFINDF        F
BSEFINDN        B
BSEFLAT         B
BSEUNFLA       27
BSENQIRE        7
```

When the builders are called

Builders are called during:
- Cold start
- Warm start
- Emergency restart

- After emergency restart
- Autoinstall logon and logoff
- APPC autoinstall
- CEDA INSTALL
- EXEC CICS CREATE
- EXEC CICS DISCARD
- Transaction routing
- Non-immediate shutdown.

Cold start

- Read information from the CSD and call builders to build RDO-defined terminals.
- Load in DFHTCT for non-VTAM terminals. Builders are not called.

Warm start

Note: A warm start is identical to an emergency restart from the builders perspective. The only difference is that Recovery Manager has no forward-recovery records to pass to DFHAPRDR.

- Read information from the global catalog and call builders to restore RDO-defined terminals.
- Load in DFHTCT for non-VTAM terminals. Builders are not called.

Emergency restart

- Read information from the global catalog and call builders to restore RDO-defined terminals.

  Note: Auto-installed terminals will not have a catalog entry if AIRDELAY=0
- Recovery Manager calls DFHAPRDR which calls the builders to restore in-flight terminals installs from the system log.
- Load in DFHTCT for non-VTAM resources. Builders are not called.

After emergency restart

Delete autoinstalled terminals after the time period has expired as specified in the AIRDELAY parameter (if the user has not logged back on before then).

APPC autoinstall

- Inquire on the model supplied by the autoinstall user program
- Install an APPC connection created from the above inquire.

Autoinstall logon and logoff

- Logon: Install terminal entry using model entry in the AMT.
- Logoff: Delete terminal entry.

CEDA INSTALL

Install VTAM terminal resources. (There is no builder process for CEDA DEFINE or ALTER.)

EXEC CICS CREATE

Install VTAM terminal resources.

EXEC CICS DISCARD

Delete VTAM terminal resources.

Transaction routing

If a TCTTE is defined as shippable, its definition is shipped to the remote system and installed there. The definition is obtained by an INQUIRE call to the builders in the Terminal Owning Region and built with an INSTALL call in the Application Owning Region.

Shutdown

Delete autoinstalled terminals from the catalog (if they had entries, and are not LU6.2 parallel connections). On a warm start, therefore, autoinstalled terminals are not recovered.

Diagnosing problems with the builders

When working on a problem associated with a builder (for example, abend or loop), it may be helpful to ask yourself the following questions:

- Why am I in a DFHBS* module? Am I doing CEDA GRPLIST install, CEDA GROUP install, autoinstall, logon, logoff, catalog, uncatalog, create or discard?
- What is the termid/sysid of the terminal I am working with (the one I am installing, deleting, cataloging or uncataloging)?
- Is this resource part of an resource definition atom?
- How is this terminal defined?
- Are there any messages associated with this terminal?

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for the DFHZCQxx modules:

- AP FCB0 - FCBF, for which the trace level is 1.

The following point IDs are provided for the DFHTBSx modules:

- AP FCC0 - FCC9, for which the trace level is 1.

The following point IDs are provided for the DFHTBSxP modules:

- AP 0630 - 0644, exception trace.
- AP FCD0 - FCD9, for which the trace level is 1.
- AP FCDA - FCDB, for which the trace level is 2.

The following point IDs are provided for the DFHTBSS module:

- AP 0620 - 0621, for which the trace level is 1.
- AP 0622 - 062E, and 0645 exception trace.

The following point IDs are provided for the DFHTONR module:

- AP 0648 - 0649, for which the trace level is 1.
- AP 064A - 064C, exception trace.

The following point IDs are provided for the DFHAPRDR module:

- AP 0601 - 0602, for which the trace level is 1.
- AP 0603 - 061E, exception trace.

The following point IDs are provided for the DFHZGTA module:

- AP FA80 - FA81, for which the trace level is 1.
- AP FA82 - FA9A, exception trace.

The following point ID is provided for message set production:

- AP FCDD, exception trace.

The following point ID is provided for DFHBSTZA:
- AP FCDE, exception trace.

See the *CICS Trace Entries* for further information.

Messages

Builder modules issue messages in the DFHZC59xx, DFHZC62xx, and DFHZC63xx series.

Message sets

If a builder finds an error, it adds a message to a message set. This set is then printed by the caller; for example:

```
DFHTCRP   Cold start (local system entry
                      and error console only)
DFHAMTP   CEDA, EXEC CICS CREATE
DFHEIQSC  EXEC CICS DISCARD CONNECTION
DFHEIQST  EXEC CICS DISCARD TERMINAL
DFHZATA   Autoinstall
DFHZATD   Autoinstall delete
DFHZATS   Install and delete transaction routed terminals
```

How messages show up in a trace

If a message is issued from a builder module (that is, those with a prefix of DFHZC59xx, DFHZC62xx, or DFHZC63xx), it appears in the trace as a table builder services message trace entry with the following point ID:
- AP FCDD, exception trace.

This trace entry is produced when a message is added to the message set and indicates there was a problem in building or deleting a terminal or connection.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 7. Built-in functions

CICS provides the application programmer with two commonly used functions: field edit and phonetic conversion.

These are functions that generally used to be coded as separate subroutines by the programmer. They are referred to as built-in functions.

The field edit function is provided by the BIF DEEDIT command of the CICS application programming interface.

The phonetic conversion function is provided as a subroutine that can be called by CICS application programs, and also by any offline programs.

Design overview

The built-in functions component includes field edit and phonetic conversion, both of which are available to a CICS application program. Also, the phonetic conversion subroutine can be used offline.

Field edit (DEEDIT)

The field edit function allows the application program to pass a field containing EBCDIC digits (0 through 9) intermixed with other values, and receive a result with all non-numeric characters removed.

For further details of this function, see the *CICS Application Programming Reference*.

Phonetic conversion

This facility allows the user to organize a file according to name (or similar alphabetic key), and access the file using search arguments that may be misspelled.

The phonetic conversion subroutine (DFHPHN) converts a name into a partial key, which can then be used to access a database file. The generated key is based upon the sound of the name. This means that names sounding similar, but spelled differently, generally produce identical keys. For example, the names SMITH, SMYTH, and SMYTHE all produce a phonetic key of S530. Likewise, the names ANDERSON, ANDRESEN, and ANDRESENN produce a phonetic key of A536. The encoding routine ignores embedded blanks in a name, so you can write names prefixed by ‘Mc’ with or without a blank between the prefix and the rest of the name, for example, ‘McEWEN’ or ‘Mc EWEN’.

For details of how to code a CALL statement for the DFHPHN subroutine according to the language of the application program, see the *CICS Application Programming Guide*.

Modules

Module	Description
DFHEBF	EXEC interface processor for BIF DEEDIT command
DFHPHN	Phonetic conversion subroutine

Exits

No global user exit points are provided for these functions.

Trace

No tracing is performed for the phonetic conversion subroutine.

The following point ID is provided for DFHEBF:
- AP 00FB, for which the trace level is BF 1.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 8. CICS-DB2 Attachment Facility

The CICS-DB2® Attachment facility allows applications programs to access and update data held in DB2 tables managed by the DB2 for OS/390 product. It also allows applications to send operator commands to a DB2 subsystem.

Design overview

The CICS-DB2 Attachment facility allows connection to a DB2 subsystem using the CICS resource manager interface (RMI), which is also known as the task related user exit interface. The Attachment facility interfaces to DB2 through a series of requests to three components of DB2, each of which processes specific types of requests:

- Subsystem Support Subcomponent (SSSC) for thread and system control requests
- Advanced Database Management Facility (ADMF) for SQL requests
- Instumentation Facility Component (IFC) for IFI requests

There no are DB2 release dependencies within the attachment facility, it can connect to a DB2 subsystem running any supported level of DB2.

The architecture of the CICS-DB2 interface is described in *CICS DB2 Guide*:

- For DB2 for OS/390 Version 6 or later, the attachment facility exploits the open transaction environment (OTE) and uses CICS-managed open TCBs.
- For DB2 for OS/390 Version 5 or earlier, the attachment facility manages its own set of subtask TCBs on which it builds threads to access DB2.

CICS Initialization

During CICS Initialization the following modules are invoked:

CICS-DB2 initialization gate DFHD2IN1

DFHD2IN1 first receives control from DFHSII1 duiring CICS initialization by means of a DFHROINM INITIALISE call. When invoked with this function DFHD2IN1 attaches a system task CSSY to run program DFHD2IN2.

DFHD2IN1 is invoked a second time later by DFHSII1 by means of a DFHROINM WAIT_FOR_INITIALIZATION call for which DFHD2IN1 issues a CICS wait to wait for DFHD2IN2 processing to complete.

CICS-DB2 recovery task DFHD2IN2

DFHD2IN2 runs under CICS system task CSSY attached by DFHD2IN1. DFHD2IN2 links to program DFHD2RP, the CICS-DB2 restart program. On return from DFHD2RP, DFHD2IN2 posts the ecb waited on by DFHD2IN1 so that CICS Initialization can continue.

CICS-DB2 restart program DFHD2RP

DFHD2RP runs under system task CSSY during CICS initialization. DFHD2RP performs the following functions:

- Adds storage manager subpools for the DFHD2ENT, DFHD2TRN and DFHD2CSB control blocks.
- Issues lock manager domain ADD_LOCK requests to add the necessary locks required by the CICS-DB2 Attachment facility to manage the dynamic chains of DFHD2LOT and DFHD2CSB control blocks, plus locks to manipulate the DFHD2GLB, DFHD2ENT and DFHD2TRN control blocks.
- Loads CICS-DB2 modules DFHD2CC, DFHD2CO, DFHD2D2, DFHD2STR, DFHD2STP and DFHD2TM
- Activates the DFHD2TM gate with the kernel.
- For cold and Initial CICS starts:

- Purges the Global catalog of DFHD2GLB, DFHD2ENT and DFHD2TRN control blocks
- For warm and emergency CICS starts:
  - Installs DFHD2GLB, DFHD2ENT and DFHD2TRN blocks found on the global catalog

CICS-DB2 Attachment startup

The CICS-DB2 Attachment facility can be started using one of the following methods:

- specifying program DFHD2CM0 in PLTPI
- specifying SIT parameter DB2CONN=YES
- Issuing the DSNC STRT command
- Issuing the CEMT or EXEC CICS SET DB2CONN CONNECTED command

All of the above ways result in an EXEC CICS SET DB2CONN CONNECTED command being issued and the CICS-DB2 startup program DFHD2STR getting control.

CICS-DB2 startup program DFHD2STR

The startup program starts by reading a temporary storage queue to obtain any parameters passed if a DSNC STRT command has been issued. It also retrieves any parameters specified via the INITPARM SIT parameter by linking to program DFHD2INI.

Next DFHD2STR must ensure the necessary DFHD2GLB block is installed. If a DFHD2GLB is already installed, representing an installed DB2CONN, then it is checked to make sure interface is currently shut before startup can proceed.

The remainder of DFHD2STR processing is as follows:

- Initialise the DFHD2GLB and set the state to 'connecting'
- MVS load the DB2 program request handler
- Attach a CICS system task to run the CICS DB2 service task CEX2
- Call DFHD2CO to connect to DB2 and obtain indoubts
- Enable the CICS-DB2 TRUE DFHD2EX1
- If connected to DB2 for OS/390 Version 5 or earlier, then issue an MVS Attach for the CICS-DB2 master subtask program DFHD2MSB and wait for DFHD2MSB initialization processing to complete
- Set the status of the connection to 'connected'
- Post CEX2 to process any indoubts passed from DB2
- Update state in the temporary storage queue to pass back to a DSNC STRT command

CICS-DB2 attachment shutdown

The CICS-DB2 Attachment facility can be stopped using one of the following methods:

- Issuing the DSNC STOP command
- Issuing the CEMT or EXEC CICS SET DB2CONN NOT CONNECTED command
- Running the CDBQ or CDBF transactions
- Shutting down CICS

All of the above ways result in an EXEC CICS SET DB2CONN NOTCONNECTED command being issued and the CICS-DB2 shutdown program DFHD2STP getting control.

CICS-DB2 shutdown program DFHD2STP

Processing in DFHD2STP is as follows:

- If CDB2SHUT is set in the dump table, take a system dump (serviceability aid)
- If a CDB2SHUT dump has not been taken, and the CICS-DB2 master subtask program DFHD2MSB has unexpectedly abended, then a system dump is taken with a dumpcode of MSBABEND.

- post CICS-DB2 service task CEX2 to terminate all subtasks then terminate itself. Wait for service task to complete.
- If present, post master subtask DFHD2MSB to terminate. Wait for it to terminate, then detach master subtask TCB.
- Call DFHD2CO to disconnect from DB2
- Call DFHD2CC to write out shutdown statistics
- If the CICS-DB2 attachment is to go into 'standbymode':
  - Re-initialise the DFHD2GLB, set the state to 'connecting'
  - Post any tasks who are waiting for shutdown to complete
  - Issues 'Waiting for DB2 attach' message
- If the CICS-DB2 attachment is not to go into 'standbymode':
  - Disable the CICS-DB2 TRUE DFHD2EX1
  - MVS delete the program request handler
  - Re-initialise the DFHD2GLB, set the state to 'shut'
  - Issue the shutdown complete message and post any tasks who are waiting for shutdown to complete

CICS-DB2 mainline processing

CICS-DB2 task related user exit (TRUE) DFHD2EX1

Control is passed to the TRUE via the CICS RMI. The TRUE manages the relationship between a CICS task (represented by a LOT control block), and a CICS-DB2 thread (represented by a CSB control block). DFHD2EX1 uses parameters set in the DB2CONN and DB2ENTRY definitions to manage use of the CICS DB2 threads, each thread running under a thread TCB.

- When connected to DB2 for OS/390 Version 5 or earlier, the thread TCB is a subtask managed by the CICS DB2 attachment facility. It is the subtask running program DFHD2EX3 which issues requests to DB2 on behalf of a CICS task. A wait/post protocol is executed between the CICS task running in the CICS-DB2 TRUE, and the subtask in program DFHD2EX3.
- When connected to DB2 for OS/390 Version 6 or later, the thread TCB is a CICS open TCB (L8 mode). Program DFHD2D2 is called under the open TCB, and issues the requests to DB2. In this case, both DFHD2EX1 and DFHD2D2 run under the L8 TCB.

The CICS-DB2 TRUE DFHD2EX1 gets invoked by the RMI for the following events:

- EXEC SQL commands and DB2 IFI commands from application programs
- syncpoint
- end of task
- INQUIRE EXITPROGRAM commands for the DB2 TRUE with the CONNECTST or QUALIFIER keywords (RMI SPI calls)
- EDF - when EDFing EXEC SQL commands
- CICS shutdown

CICS-DB2 coordinator program DFHD2CO

The coordinator program runs under the CICS Resource owning (RO) TCB, and handles the overall connection between CICS and a DB2 subsystem. It is called :

- by DFHD2STR during startup of the attachment facility to issue the coordinator identify to DB2, that is to establish connection to DB2. Once established, it passes DB2 an ECB to be posted should DB2 terminate, and it also obtains from DB2 a list of units of work (UOWs) that DB2 is indoubt about. This list is anchored off the CICS-DB2 global block (DFHD2GLB) for processing later in startup.
- by DFHD2STP during shutdown of the attachment facility to terminate the identify to DB2 and so disconnect.

- by the CICS-DB2 TRUE DFHD2EX1 during resync processing to pass the resolution of a indoubt unit of work to DB2. Indoubt resolution has to be done under the same TCB that issued the coordinator identify to DB2.

CICS-DB2 master subtask program DFHD2MSB

When operating with DB2 for OS/390 Version 5 or earlier, the DFHD2MSB TCB is attached by DFHD2STR during startup of the Attachment facility. It runs as a 'daughter' of the main CICS TCB. It is 'mother' to all the subtask TCBs which process the DB2 work. The DFHD2MSB TCB is detached by DFHD2STP during CICS-DB2 Attachment shutdown.

The main functions of DFHD2MSB are:

- To attach thread subtasks as required
- To detach thread subtasks as required
- To provide a recovery routine to cleanup if a thread subtask fails

CICS-DB2 subtask program DFHD2EX3

When operating with DB2 for OS/390 Version 5 or earlier, a CICS-DB2 subtask TCB is attached by DFHD2MSB when required by DFHD2EX1. It runs as a daughter of the DFHD2MSB TCB and a granddaughter of the main CICS TCB. A CICS-DB2 subtask TCB normally remains active for the lifetime of the CICS Attachment facility and terminates as part of CICS-DB2 Attachment facility shutdown. Exception conditions that cause a subtask TCB to be detached are:

- if the DB2CONN TCBLIMIT parameter is lowered
- if a CICS task is forcepurged whilst its associated subtask is active in DB2
- If a failure occurs during syncpoint processing during the indoubt window requiring the thread to be released.

The DFHD2EX3 program issues requests to DB2 using the DB2 SSSC, ADMF and IFC interfaces communicating via the DB2 program request handler DSNAPRH. In order to process DB2 requests a TCB first has to IDENTIFY to DB2, secondly it has to SIGNON to DB2 to establish authorization ids to DB2. Thirdly a thread has to be created. Once a thread has been created API and syncpoint requests can flow to DB2. Subsequent SIGNON requests can occur for a thread to change authorization ids to DB2 or for the purposes of DB2 cutting accounting records (partial SIGNON) When a thread is nolonger required it is terminated. The TCB remains identified and signed on to DB2 and awaits another request requiring it to create a thread again.

Each DB2 subtask runs an instance of program DFHD2EX3 and each is represented by a DFHD2CSB control block. A CSB control block is anchored to one of three CSB chains depending on its state (an active thread within a UOW, a thread waiting for work, or an identified, signed on TCB with no thread). The CICS-DB2 TRUE DFHD2EX1 manages the CSB chains.

CICS-DB2 thread processor DFHD2D2

The thread processor program DFHD2D2 is used only when operating with DB2 for OS/390 Version 6 and above, when the CICS-DB2 Attachment Facility uses CICS open TCBs (L8 TCBs) rather than privately managed subtask TCBs. In the Open Transaction environment (OTE), the CICS-DB2 TRUE DFHD2EX1 is invoked under an L8 TCB. Instead of posting a subtask, DFHD2EX1 calls DFHD2D2 under the L8 TCB. DFHD2D2 performs the same functions as performed by subtask program DFHD2EX3 in a non OTE environment, that is issuing the identify, signon, create thread, terminate thread calls to DB2, plus the api and syncpoint calls to DB2.

DFHD2D2 is called via a subroutine domain call on which the address of the relevant connection control block (DFHD2CSB) is passed. On the first call of a unit of work, DB2 is called to "associate" the connection with the calling L8 TCB. Once this is done, calls to DB2 can proceed as normal. When a DB2 thread is released from a CICS transaction (typically at syncpoint), the connection is "dissociated" from the L8 TCB. Hence a connection control block (DFHD2CSB) has an affinity to an L8 TCB whilst is associated. With DB2 for OS/390 Version 5 and below a connection has a permanent affinity to its subtask TCB.

CICS-DB2 service task program DFHD2EX2

The CICS-DB2 service task program DFHD2EX2 runs as a CICS system task under transaction CEX2. Its mains functions are:

- To wait for DB2 to startup if DB2 is down when connection is attempted if STANDBYMODE=RECONNECT or CONNECT is specified in the DB2CONN.
- To initiate shutdown of the CICS-DB2 Attachment facility if posted to do so.
- To perform the protected thread purge cycle.
- To issue EXEC CICS RESYNC to process DB2 indoubts.
- For DB2 for OS/390 Version 5 or earlier, to terminate all subtasks during CICS-DB2 Attachment facility shutdown.

CICS-DB2 PLTPI program DFHD2CM0

Used in PLTPI or as a result of DB2CONN=YES being set in the SIT. It issues an EXEC CICS SET DB2CONN CONNECTED command to start up the CICS DB2 Attachment facility.

CICS-DB2 comand processor DFHD2CM1

DFHD2CM1 processes commands issues via the DSNC command. The following commands are processed:

- DSNC STRT - EXEC CICS SET DB2CONN CONNECTED command issued
- DSNC STOP - EXEC CICS SET DB2CONN NOTCONNECTED command issued
- DSNC MODIFY DEST - EXEC CICS SET DB2CONN MSGQUEUEn command issued
- DSNC MODIFY TRAN - EXEC CICS SET DB2CONN THREADLIMIT or EXEC CICS SET DB2ENTRY THREADLIMIT command issued.
- DSNC DISC - call passed to DFHD2CC to disconnect threads
- DSNC DISP PLAN - call passed to DFHD2CC to display information on threads for a particular DB2 plan
- DSNC DISP TRAN - call passed to DFHD2CC to display information on threads for a transaction.
- DSNC DISP STAT - call passed to DFHD2CC to write out statistics
- DSNC -db2command - DB2 IFI ccommand issued to send operator command to the connected DB2 subsystem.

CICS-DB2 shutdown quiesce program DFHD2CM2

Runs under transaction CDBQ. Issues an EXEC CICS SET DB2CONN NOTCONNECTED WAIT command to shutdown the CICS-DB2 Attachment facility.

CICS-DB2 shutdown force program DFHD2CM3

Runs under transaction CDBF. Issues an EXEC CICS SET DB2CONN NOTCONNECTED FORCE command to shutdown the CICS-DB2 Attachment facility.

CICS-DB2 table manager DFHD2TM

Handles installs, discards, inquire and set requests for the DFHD2GLB, DFHD2ENT and DFHD2TRN control blocks representing the DB2CONN, DB2ENTRY and DB2TRAN resources. Callers of DFHD2TM are:

- DFHAMD2 - for CEDA install and EXEC CICS CREATE
- DFHD2EX1 - to complete disablement of a DB2ENTRY or to complete Attachment facility shutdown
- DFHD2RP - to install objects from the Global Catalog during CICS restart
- DFHEIQD2 - for EXEC CICS INQUIRE,SET and DISCARD of DB2 objects
- DFHESE - for inquiry during EXEC CICS QUERY SECURITY processing.

CICS DB2 statistics program DFHD2ST

Called by AP domain statistics program DFHAPST to process CICS-DB2 statistics for EXEC CICS COLLECT STATISTICS and EXEC CICS PERFORM STATISTICS commands.

CICS DB2 connection control program DFHD2CC

DFHD2CC proceses the following requests:

- Start_db2_attachment - request routed on to DFHD2STR
- Stop_db2_attachment - request routed on to DFHD2STP
- Write_db2_statistics - statistics collected from control blocks and are written out to the terminal, to transient data or to SMF.
- disconnect_threads - CSB control blocks searched and marked so that threads are terminated when they are next released.
- display_plan and display_tran - thread information collected from control blocks and output to the terminal.

CICS DB2 EDF processor DFHD2EDF

Receives control from CICS-DB2 TRUE DFHD2EX1 when the TRUE is invoked for an EDF request. DFHD2EDF uses the RMI provided parameters to format the screen to be output by EDF before and after an EXEC SQL request is issued.

Control blocks

DFHD2SS (CICS-DB2 static storage)

CICS-DB2 static storage (D2SS) is acquired by DFHSIB1 and anchored off field SSZDB2 in the static storage address list DFHSSADS. The static storage is initialized by the CICS-DB2 restart program DFHD2RP. Its lifetime is that of the CICS region. CICS-DB2 static storage holds information such as storage manager, lock manager and directory manager tokens acquired during restart processing before any other CICS-DB2 control blocks are installed.

DFHD2GLB (CICS-DB2 global block)

The DFHD2GLB block represents an installed DB2CONN definition. It is getmained by DFHD2TM when a DB2CONN is installed and freemained by DFHD2TM when a DB2CONN is discarded. It holds CICS-DB2 state data global to the connection and also the state data for pool threads and commands threads. The pool and command sections of the DFHD2GLB are mapped by a common type definition DFHD2RCT which is also used to map the DFHD2ENT control block.

The DFHD2GLB block is anchored off CICS-DB2 static storage in field D2S_DFHD2GLB.

DFHD2ENT (CICS-DB2 DB2ENTRY block)

The DFHD2ENT block represents an installed DB2ENTRY definition. It is getmained by DFHD2TM when a DB2ENTRY is installed and freemained by DFHD2TM when a DB2ENTRY is discarded. It uses a type definition DFHD2RCT in common with the pool and command sections of the DFHD2GLB block to achieve a common layout for all three areas. A DFHD2ENT block is located using a directory manager index that is keyed off the RDO name of the DB2ENTRY.

DFHD2TRN (CICS-DB2 DB2TRAN block)

The DFHD2TRN block represents an installed DB2TRAN definition. It is getmained by DFHD2TM when a DB2TRAN is installed and freemained by DFHD2TM when a DB2TRAN is discarded. A DB2TRAN can be located in two ways. Firstly by a directory manager index keyed off the RDO name of the DB2TRAN. Secondly by a directory manager index keyed off the transaction id associated with the DB2TRAN.

DFHD2CSB (CICS-DB2 connection block)

The DFHD2CSB block represents a CICS-DB2 connection, with or without a thread. A DFHD2CSB is created by DFHD2EX1 prior being passed to DFHD2EX3 or DFHD2D2. A DFHD2CSB is freed by DFHD2EX1 after the DFHD2EX3 program has returned to MVS, or when DFHD2D2 indicates it should be freed. A DFHD2EX3 block is anchored off one of several CSB chains from a DB2ENTRY or the DFHD2GLB depending on the state of the connection and the DB2 thread.

DFHD2GWA (CICS-DB2 global work area)

The DFHD2GWA block is the global work area of the CICS-DB2 task related user exit (TRUE) DFHD2EX1. It is getmained when the TRUE is enabled, and freemained when the TRUE is disabled. The D2GWA holds a chain of LOT control blocks representing the tasks currently using the CICS-DB2 interface.

DFHD2LOT (CICS-DB2 life of task block)

The DFHD2LOT block is the task local work area of the CICS-DB2 task related user exit (TRUE) DFHD2EX1. It is getmained by DFHERM when a task first calls the CICS-DB2 TRUE. It is freemained by DFHERM at end of task. Its address is passed to DFHD2EX1 by DFHERM in parameter UEPTAA in the DFHUEPAR RMI parameter list.

The DFHD2LOT holds CICS-DB2 state information for a CICS task using the CICS-DB2 interface.

Modules

Module	Description
DFHD2CC	CICS-DB2 connection control program
DFHD2CO	CICS-DB2 coordinator program
DFHD2CM0	CICS-DB2 PLTPI startup program
DFHD2CM1	CICS-DB2 command processor
DFHD2CM2	CICS-DB2 quiesce shutdown program
DFHD2CM3	CICS-DB2 force shutdown program
DFHD2D2	CICS-DB2 thread processor
DFHD2EDF	CICS-DB2 EDF processor
DFHD2EX1	CICS-DB2 task related user exit (TRUE)
DFHD2EX2	CICS-DB2 service task program
DFHD2EX3	CICS-DB2 subtask program
DFHD2INI	CICS-DB2 Initparm processor
DFHD2IN1	CICS-DB2 initialization gate
DFHD2IN2	CICS-DB2 recovery task
DFHD2MSB	CICS-DB2 master subtask program
DFHD2RP	CICS-DB2 restart program
DFHD2STP	CICS-DB2 shutdown program
DFHD2STR	CICS-DB2 startup program
DFHD2ST	CICS-DB2 statistics program
DFHD2TM	CICS-DB2 table manager
DSNCUEXT	CICS-DB2 sample dynamic plan exit

Exits

There are no Global user exits provided by the CICS DB2 Interface.

The CICS DB2 interface does however provide a dynamic plan 'exit' in the form of a user-replaceable program. A sample default exit is provided called DSNCUEXT. A dynamic plan exit allows the name of the plan to chosen dynamically at execution time. For further information about dynamic plan exits see the CICS DB2 Guide.

Trace

The CICS-DB2 Attachment facility outputs trace entries in the range AP 3100 to AP 33FF. Trace output from the CICS-DB2 TRUE (DFHD2EX1) and the thread processor (DFHD2D2), and GTF trace from the CICS-DB2 subtask is controlled by the RI (RMI) trace flag. Trace from the rest of the attachment and other CICS-DB2 modules is controlled by the FC (File Control) trace flag.

Statistics

A limited set of CICS-DB2 statistics can be obtained by issuing the DSNC DISP STAT command, which will output the statistics to a CICS terminal. The same format of statistics is output to a nominated transient data queue when the CICS-DB2 Attachment facility is shut down For more information see the *CICS DB2 Guide*.

A more comprehensive set of CICS-DB2 statistics can be obtained by issuing an EXEC CICS PERFORM STATISTICS RECORD command with the DB2 keyword, or by issuing the EXEC CICS COLLECT STATISTICS command with the DB2CONN or DB2ENTRY keywords. CICS-DB2 Global statistics are mapped by DSECT DFHD2GDS. CICS-DB2 resource statistics are mapped by DSECT DFHD2RDS. For more information see the *CICS Performance Guide*.

Chapter 9. Command interpreter

The command interpreter demonstrates to the application programmer the syntax of CICS commands and the effects of their execution. It can also be used to perform simple one-off tasks whose nature does not justify the writing of a permanent application.

Design overview

The command interpreter is invoked by the CECI transaction and is an interactive, display-oriented tool that checks the syntax of CICS commands and executes them. Another transaction, CECS, performs only syntax checking.

The user enters a command that is analyzed in the same way as it would be by the command translator, which processes it as if it were part of an application program. The results of this analysis, including any messages, an indication of defaults assumed, and the entire syntax of the command, are then displayed.

When the command is syntactically valid, the user can request its execution. The interpreter calls DFHEIP, passing a parameter list precisely as would be passed during the execution of a program that contained the command.

The interpreter does all this using the same command-language tables as are used by the command translator. These tables contain data that define the syntax of CICS commands and the contents of the parameter lists required by DFHEIP to execute them.

Modules

Module	Function
DFHECIP	Invoked by CECI. Checks that the terminal is suitable. Obtains and initializes working storage. Loads the language tables. Links to DFHECID
DFHECSP	Same as DFHECIP, but invoked by CECS
DFHECID	Receives data from the terminal and sends back a display. Analyzes commands. Constructs parameter lists for DFHEIP, which it calls. Deals with PF keys
DFHEITAB	Command-language table (application programmer commands)
DFHEITBS	Command-language table (system programmer commands).

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

Chapter 10. CSD utility program (DFHCSDUP)

The CSD utility program, DFHCSDUP, provides offline services for you to list and modify the resource definitions in the CICS system definition (CSD) file. DFHCSDUP can be invoked as a batch program, or from a user-written program running either in batch mode or under TSO. The second method provides a more flexible interface to the utility, allowing for the specification of up to five user exit routines to be called at various points during DFHCSDUP processing.

Further information about using DFHCSDUP is given in *CICS Operations and Utilities Guide*and *CICS Customization Guide*.

The following commands can be used with DFHCSDUP:

```
ADD
ALTER
APPEND
COPY
DEFINE
DELETE
EXTRACT
INITIALIZE
LIST
MIGRATE
PROCESS
REMOVE
SCAN
SERVICE
UPGRADE
USERDEFINE
VERIFY
```

These commands are described in the *CICS Operations and Utilities Guide*.

Design overview

When DFHCSDUP is invoked, control passes to the utility command processor (DFHCUCP), which validates commands and invokes the appropriate routine to execute the requested function. Unless DFHCSDUP has been invoked from a user program specifying a get-command exit, DFHCUCP takes a command from the input data set, using DFHCUCB to obtain the command and DFHCUCAB to analyze and parameterize it. When supplied, the get-command exit is invoked from the point during DFHCUCB's processing where commands would otherwise be read from SYSIN (or an alternatively named input data set when DFHCSDUP is invoked from a user program).

Some syntax errors are diagnosed and reported by DFHCUCAB, and further contextual validation takes place in DFHCUCV. Valid commands are then passed to the relevant service routine for execution; for example, a MIGRATE command is handled by DFHCUMIG. If command execution is successful, the next command is processed.

All commands are validated, but the execution of commands from the input data set stops when an incorrect command is encountered, and execution of subsequent commands is also suppressed if an error of severity 8 or higher occurs when the command is executed. When commands are supplied by a get-command exit, however, DFHCSDUP attempts to execute all commands, even if an error is detected in the command syntax or during processing (unless the error is serious enough to warrant an ABEND).

If errors occur while processing commands, error messages in the DFH51xx, DFH52xx, DFH55xx, and DFH56xx series are written to SYSPRINT (or an alternatively named output data set when DFHCSDUP is invoked from a user program).

An ESTAE environment is established by DFHCUCP shortly after the start of DFHCSDUP processing. If an operating system abend subsequently occurs, control passes to the ESTAE exit routine, which then returns to MVS requesting a dump and scheduling a retry routine to get control. This retry routine attempts cleanup processing before returning to the caller of DFHCSDUP with a return code of ‘16’.

To protect the integrity of the CSD, DFHCUCP issues a STAX macro to defer the handling of any attention interrupts that may occur in a TSO environment until all processing associated with the current command has been completed.

DFHCSDUP uses batch versions of RDO routines from the parameter utility program (DFHPUP) and the CSD management program (DFHDMP) to read, write, and update resource definitions on the CSD file. All CSD control functions use the batch environment adapter (DFHDMPBA), which performs environment-dependent VSAM operations on the CSD file. DFHDMPBA also processes all interactions with operating system services.

Modules

DFHCSDUP is link-edited from a number of object modules, including batch versions of routines from DFHPUP and DFHDMP.

Exits

When invoked as a conventional batch program, DFHCSDUP supports only one user exit: the EXTRACT exit, which is invoked at various stages during the processing of an EXTRACT command. The name of the user-written program to get control must be specified by the USERPROGRAM keyword of the EXTRACT command. Details of selected CSD objects are passed to the user exit program so that users can analyze the contents of their CSD in any way they may choose.

When invoked from a user program, DFHCSDUP supports the following five user exits, the addresses of which can be specified in the EXITS parameter of DFHCSDUP’s entry linkage:
1. Initialization exit—invoked by DFHCUCP
2. Termination exit—invoked by DFHCUCP
3. EXTRACT exit—invoked by DFHCULIS
4. Get-command exit—invoked by DFHCUCB
5. Put-message exit—invoked by DFHBEP.

Note: A user exit routine specified by the USERPROGRAM keyword of an EXTRACT command is used in preference to any EXTRACT exit routine specified on the entry linkage.

For further information about these user exits, see the *CICS Customization Guide*.

Trace

Trace points are not applicable to offline utilities.

Statistics

The following statistics are maintained by DFHCSDUP, and are written, when appropriate, to SYSPRINT (or alternatively named output data set):

```
CMDSEXOK   Commands executed OK
CMDSINER   Commands in error
CMDSNOTX   Commands not executed
CMDSWARN   Commands with warning messages.
```

All the above statistics are kept in DFHCUCP’s static storage and are always output at the end of processing.

All the following statistics are kept in DFHCUMIG's static storage and the appropriate statistics are also output to SYSPRINT (or its replacement). For example, if a user migrates an FCT, only TOTFILE and TOTLSRP are output.

```
TOTCONS    Total connections migrated
TOTFILE    Total files migrated
TOTLSRP    Total LSR pools migrated
TOTMAPS    Total map sets migrated
TOTPGMS    Total programs migrated
TOTPRFG    Total profiles generated
TOTPRFM    Total profiles migrated
TOTPSTS    Total partition sets migrated
TOTSESS    Total sessions migrated
TOTTRAS    Total transactions migrated
TOTTRMS    Total terminals migrated
TOTTYPS    Total typeterms migrated
```

Chapter 11. Database control (DBCTL)

An overall description of DL/I database support is given in Chapter 15, “DL/I database support,” on page 119. This section gives information that is specific to database control (DBCTL).

Design overview

The CICS support that enables connection to DBCTL, via the database resource adapter (DRA), is based on the CICS resource manager interface (RMI), also known as the task-related user exit interface. However, because it is necessary to provide compatibility with the existing CICS-DL/I implementation (in terms of link-edit stubs, API return codes, and so on), a limited amount of support within CICS itself is provided, but there are no DBCTL release dependencies within the CICS modules.

The main components of the CICS-DBCTL interface are shown in Figure 31:

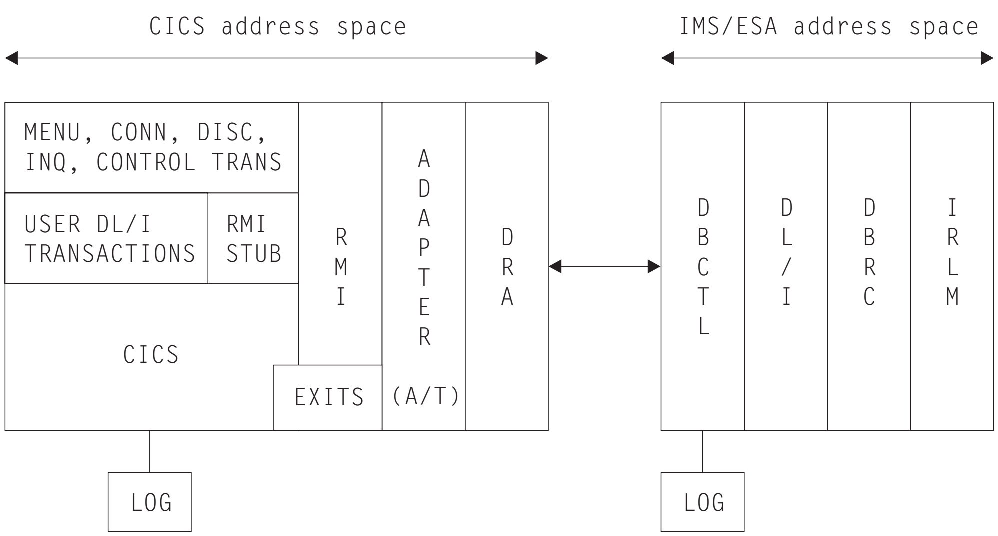


Figure 31. The major components of the CICS-DBCTL interface

- The connection process (CICS-DBCTL)

CICS-DBCTL connection and disconnection programs
These programs are used for establishing and terminating the connection with the DRA.

CICS-DBCTL control program
This program is responsible for resolving in-doubt units of work after a CICS or DBCTL failure. It also outputs messages when DBCTL notifies CICS of a change in the status of the CICS-DBCTL interface.

When the CICS disconnects from DBCTL, the control program is responsible for invoking the disable program which performs cleanup.

DRA control exit
This exit is invoked by the DRA, when connection has been established with the DBCTL address space, to initiate the resynchronization process, that is, to initiate the resolution of in-doubt units of work. It is also invoked to handle cases where connection to DBCTL cannot be achieved or when the connection has failed.

DBCTL user-replaceable program
This program is invoked whenever CICS successfully connects to DBCTL and whenever CICS disconnects from DBCTL.

Disable program
This program is invoked when CICS disconnects from DBCTL.

- The DBCTL call processor program

  The function of this program is to issue an RMI call to DBCTL and to maintain compatibility with the existing CICS-DL/I interface in areas such as application program return codes, and so on.
- The interface layer

  The adapter
  The adapter's primary responsibility is interfacing the RMI and DRA parameter lists. Other responsibilities include the issuing of DRA initialization and termination calls, when invoked by the CICS connection and disconnection programs, and the management of CICS tasks, in order to effect an orderly shutdown of the CICS-DBCTL interface.

  DRA suspend and resume exits
  These exits are invoked by the DRA in order to suspend and resume a CICS task while a DL/I call is processed by DBCTL.

  Adapter exits
  There are four exits for use by the adapter:
  - The statistics exit
  - The token exit
  - The monitoring exit
  - The status exit.

Details of these components are described in the following sections.

Note: CICS documentation uses the term “connecting and disconnecting from DBCTL”. The DRA documentation refers to “initializing and terminating the CICS-DBCTL interface”. In general, these two terms are synonymous.

The connection process

Connection and disconnection programs

In order to initialize, terminate, and inquire on the status of the interface, a set of four programs is available:
1. Menu program
2. Connection program
3. Disconnection program
4. Inquiry program.

Menu program (DFHDBME): This permits a terminal user to display a menu, which offers the option of connecting and disconnecting from DBCTL.

The menu program passes control to either the connection or the disconnection program, as appropriate, using the COMMAREA to pass any overrides and parameters.

In the case of connection, it offers the ability to supply the suffix of the DRA startup parameter table and the name of the DBCTL region. The DRA startup parameter table contains various parameters, mostly relating to the initialization of the CICS-DBCTL interface, including the name of the DBCTL region and the minimum and maximum number of CICS-DBCTL threads. It also contains the length of time in seconds that the DRA waits after an unsuccessful attempt to connect to DBCTL, before attempting to connect again.

For disconnection, it offers the ability to specify whether an orderly or immediate disconnection from DBCTL is required.

The menu program is intended for use by CICS operators or network controllers, that is, users with special privileges.

BMS maps are used for both the menu and the inquiry programs. It should be noted that the bottom half of the menu screen includes all the items which appear on the inquiry screen, and the values are displayed on entry to the menu program, if they are known. The DRA startup table suffix is not included on the inquiry screen because the DRA startup table contains the application group name which is used for security checking.

After a connection request has been issued, it is possible to issue a disconnection request (orderly or immediate) from the menu program while the connection process is still in progress. After an orderly disconnection request has been issued, it is also possible to issue an immediate disconnection request while the orderly disconnection process is in progress. This has the effect of upgrading the orderly disconnection to an immediate disconnection.

Connection program (DFHDBCON): This program invokes the adapter requesting connection to DBCTL.

This program can be invoked either from the menu program or from the CICS PLT. It issues an ATTACH request of the CICS control program that later carries out resynchronization of in-doubt units of work with DBCTL. The control program then issues a WAIT request.

The connection program continues by loading, activating (using the EXEC CICS ENABLE command), and then calling the adapter (using a DFHRMCAL request). A set of parameters is passed to the adapter which includes:
- The CICS applid
- The DRA startup parameter table suffix (optional)
- The DBCTL ID (optional)
- A set of exit addresses.

As a result of the DFHRMCAL request issued from the connection program, the adapter loads the DRA startup/router module from the CICS STEPLIB library and passes control to it, supplying it with various parameters including the CICS applid, DRA startup parameter table suffix, and DBCTL ID. The DRA startup/router module loads the DRA startup table. It then initiates the processes required to establish the DRA and then returns control to the adapter which, in turn, returns control to the connection program which then terminates. Until this point is reached, any DBCTL requests issued from CICS tasks are rejected by the CICS RMI stub (the DBCTL call processor).

The DRA startup/router module is responsible for establishing the DRA environment, using the parameters specified in the DRA startup table in the CICS STEPLIB library, overridden by any parameters passed to it.

The DRA establishes contact with the DBCTL address space and then invokes the control exit to initiate the resynchronization process.

Disconnection program (DFHDBDSC): This program invokes the adapter requesting disconnection from DBCTL.

The disconnection program is used to terminate the DRA environment. Two types of disconnection are available:

Orderly disconnection
All existing CICS tasks using DBCTL are allowed to run to completion.

Immediate disconnection
Existing DL/I requests are allowed to complete but no further DL/I requests are accepted.

In both cases a DBCTL U113 abend is avoided. (DBCTL can issue a U113 abend if CICS terminates while there is an active DL/I thread running on its behalf in DBCTL. The thread remains active for the duration of the PSB schedule, but DBCTL would issue a U113 abend if the thread is doing something for the CICS task.)

The disconnection program calls the adapter, using DFHRMCAL, supplying a parameter to indicate the type of termination required.

In the case of immediate disconnection, the adapter issues a DRA TERM call and returns to the disconnection program only when all existing DL/I threads have completed. In the case of orderly disconnection, the adapter assumes responsibility for managing CICS tasks, that is, it continues to accept requests for current tasks using DBCTL until they terminate, but does not allow new CICS tasks to use DBCTL. When the adapter detects that the count of permitted tasks has reached zero, it issues a DRA TERM call.

The disconnection program finally posts the control program to notify it of the fact that the CICS-DBCTL interface has been terminated. The control program then terminates after starting the disable program. The disable program issues a DISABLE command for the adapter, and performs cleanup.

It should be noted that the terminal used to invoke the disconnection program is released after the input to the menu screen has been validated, enabling the terminal operator to use other programs. Any further messages from the disconnection process are generated centrally.

Inquiry program (DFHDBIQ): This program enables the user to inquire on the status of the interface. It is intended for a wider audience than the menu program; for example, application programmers.

Control program (DFHDBCT)

The control program is invoked in the following circumstances:

- When the control exit is invoked by the adapter on behalf of the DRA
- When a CEMT FORCEPURGE command is issued for a CICS task executing in DBCTL
- When the disconnection program has received a response from the adapter as a result of a CICS-DBCTL interface termination request.

Its function in all cases is to issue messages. It then issues a WAIT after every invocation. Also, it has some special functions in three cases:

1. When contact has been made with DBCTL and resynchronization of in-doubts is required.

   In this case, the control program issues the command:

   ```
   EXEC CICS RESYNC ENTRYNAME(adapter)
   IDLIST(DBCTL's in-doubts) ...
   ```

   This causes CICS to create tasks for each in-doubt unit of work. Each task performs resynchronization and then informs the adapter via the CICS syncpoint manager as to whether the task has committed or backed out. The adapter then notifies the DRA on a task basis.

   The following is a list of the possible calls to the adapter from the CICS syncpoint manager:
   - Prepare to commit
   - Commit unconditionally[1]
   - Backout[1]
   - Unit of recovery is lost to CICS cold start[2]
   - DBCTL should not be in-doubt about this unit of recovery[2].

   Notes:

   [1] These items can be issued as a result of a RESYNC request.

   [2] These items can be issued as a result of a RESYNC request only.

2. When /CHECKPOINT FREEZE has been requested.

   In this case, the control program invokes the disconnection program requesting an orderly disconnection from DBCTL. Generally, an orderly disconnection from DBCTL allows CICS tasks already using DBCTL to continue until task termination. However, when a /CHECKPOINT FREEZE has been requested, DBCTL prevents any PSB schedules from taking place. Thus, in this case, some tasks might be terminated before end of task is reached with a ‘DBCTL not available’ return code, if they issue a subsequent PSB schedule request.

3. When the disconnection program invokes the control program.

   In this case, the control program starts the disable program.

DRA control exit (DFHDBCTX)

The control exit is invoked in the DRA environment in the following circumstances:

- When contact has been established with the DBCTL address space, in order to initiate resynchronization.

  The control exit is invoked in the DRA environment whenever contact has been established with DBCTL, whether invoked by the user or due to the DRA automatically reestablishing contact after a DBCTL failure. The control exit receives an input parameter list that includes the DBCTL ID, DBCTL’s list of in-doubt units of work, and the DBCTL RSE name. The control exit posts the control program, which actually performs the resynchronization.
- When the MVS subsystem interface (SSI) rejects the IDENTIFY request to DBCTL, thereby causing the IDENTIFY to fail.

  This could occur if the DRA was trying to issue an IDENTIFY request to a DBCTL subsystem that was not running. In this case the control exit sets a response code of ‘0’. The first time in a connection attempt that the DRA receives a ‘0’ response after an MVS SSI failure, the DRA outputs message DFS690A inviting the operator to reply WAIT or CANCEL. On subsequent failures when a response code of ‘0’ is returned, the DRA waits for the length of time specified in the DRA startup table before attempting the IDENTIFY request again.
- When DBCTL rejects the IDENTIFY request to DBCTL; for example, incorrect application group name (AGN) supplied.

  In this case, the control exit asks the DRA to terminate.
- When the operator replies CANCEL to the DFS690A message during DRA initialization, because contact cannot be established with DBCTL.

  In this case, the control exit notifies the DRA to terminate immediately.
- When DBCTL abnormally terminates.

  In this case, the control exit invokes the control program and then it asks the DRA to issue an IDENTIFY request to DBCTL.
- When the DRA abnormally terminates.

  In this case, it is not possible to access DBCTL from the same CICS session without initializing the CICS-DBCTL interface using the menu program.
- When a /CHECKPOINT FREEZE request has been issued to DBCTL.

  Note that /CHECKPOINT FREEZE is the command used to close down a DBCTL subsystem. In this case the control exit invokes the control program which, in turn, invokes the disconnection program requesting an orderly disconnection from DBCTL. The control exit notifies the DRA to wait for a termination request.

DBCTL user-replaceable program (DFHDBUEX)

The DBCTL user-replaceable program, DFHDBUEX, is invoked whenever CICS successfully connects or disconnects from DBCTL. It provides the opportunity for the customer to supply code to enable and disable CICS-DBCTL transactions at these times.

The program runs as a CICS application and can thus issue EXEC CICS requests. The program is invoked with a CICS COMMAREA containing the following parameters:

- Request type: CONNECT | DISCONNECT
- Reason for disconnection: MENU DISCONNECTION | /CHECKPOINT FREEZE | DRA FAILURE | DBCTL FAILURE

- DRA startup table suffix
- DBCTL ID.

See the *CICS Customization Guide* for information about the DFHDBUEX program.

Disable program (DFHDBDI)

The disable program, DFHDBDI, is invoked when CICS disconnects from DBCTL. It performs cleanup, which includes disabling the adapter.

The DBCTL call processor program (DFHDLIDP)

Among the functions of the DBCTL call processor program, DFHDLIDP, are:

Issuing DFHRMCAL requests to the adapter: DL/I requests issued from application programs that have been routed to this module are passed on to the adapter. The DBCTL call processor constructs a register 1 parameter list that includes the DL/I parameter list and a thread token. It then issues a DFHRMCAL request.

It is the responsibility of this module to generate the thread token required by the DRA.

Maintaining return code compatibility: If any calls are made to the RMI before the first part of the connection process has completed, that is, before the DFHDBCON program has received a “successful” response code from the DRA via the adapter, error return codes are set in the task control area (TCA) to indicate that DBCTL is unavailable. These codes are put in the user interface block (UIB) by the DL/I call router program, DFHDLI.

Similarly, the DBCTL call processor informs application programs when DBCTL is no longer available; for example, after a DBCTL abend.

Another function of the call processor is to set up the TCA fields, TCADLRC and TCADLTR, with response and reason codes respectively for the call. This ensures that the application program continues to receive responses indicating normal response, NOTOPEN, and INVREQ conditions, with the appropriate response and reason codes in the corresponding UIB fields, UIBFCTR and UIBDLTR, after NOTOPEN and INVREQ conditions have been raised.

Initiating PC abends: If an ‘unsuccessful’ return code is passed back to CICS as a result of a DBCTL request, indicating that the CICS thread must be abended, the DBCTL call processor issues a PC ABEND, which invokes syncpoint processing to back out changes made to recoverable resources. Various abend codes can be issued. Note that, in the case of a deadlock abend (abend code ADCD) it may be possible to restart the program.

Exception trace entries are output in the case of transaction abends.

Writing CICS messages: For any thread abend in DBCTL, a CICS message is written indicating the abend code passed back to CICS in the field PAPLRETC. Similarly, for any scheduling failures, where the application program receives the UIBRCODE field (UIBFCTR and UIBDLTR fields combined) set to X'0805', the scheduling failure subcode is contained in a CICS message.

The interface layer

Adapter (DFHDBAT)

Control is passed to the adapter via the CICS RMI. It is the responsibility of the adapter to construct the DRA INIT, DRA TERM, and DRA THREAD parameter lists from the RMI parameter list passed to it. It must also transform the DRA parameter list passed back after a DL/I call to the format expected by CICS.

Part of the DRA parameter list requires two tokens to be generated by CICS:
1. A thread token

2. A recovery token.

The thread token is generated by the DBCTL call processor, and enables a CICS unit of work to be related to a DBCTL unit of work. It is used by the asynchronous RESUME exit to identify the CICS thread to be resumed after a DL/I call.

The 16-byte recovery token is constructed by concatenating an 8-byte unique CICS subsystem name (the CICS applid) with the 8-byte CICS RMI recovery token (also known as the unit of work ID).

A further responsibility of the adapter is to manage CICS tasks when an orderly termination of the CICS-DRA interface has been requested by means of the CICS termination program. In this case, it continues to accept DL/I requests from CICS tasks currently using DBCTL, but does not allow new CICS tasks to use DBCTL. When the adapter detects that the count of current tasks has reached zero, it issues a DRA TERM call to shut down the interface.

Table 3 summarizes the types of invocations of the adapter code from CICS, and how the adapter reacts to the individual invocation.

Table 4 summarizes the types of invocations of the adapter code from the DRA, and how the adapter reacts to each individual invocation.

Table 5 on page 100 summarizes the cases when the adapter invokes the adapter exits.

Table 3. CICS-adapter request summary

Invocation	Invoker	Adapter action
Initialize	Connection program	Issues DRA INIT
Terminate-Orderly	Disconnection program	Issues DRA TERM after waiting for CICS-DBCTL tasks to quiesce
Terminate-Fast	Disconnection program	Issues DRA TERM
PSB Schedule	DBCTL call processor	Issues THREAD SCHED
DL/I request	DBCTL call processor	Issues THREAD DLI
Prepare	CICS syncpoint manager	Issues THREAD PREP
Commit	CICS syncpoint manager	Issues THREAD COMTERM
Abort	CICS syncpoint manager	Issues THREAD ABTTERM
Lost To CICS cold start	CICS syncpoint manager	Issues COLD request
DBCTL should not be in doubt	CICS syncpoint manager	Issues UNKNOWN request
Task is terminating	CICS task manager	Issues TERMTHRD
Force Purge Task	Control program	Issues PURGE THREAD
Orderly CICS Term	CICS termination	Issues DRA TERM after waiting for CICS-DBCTL tasks to quiesce
Immediate CICS Term	CICS termination	Issues DRA TERM
CICS is abending	CICS termination	Issues DRA TERM
CICS has been canceled	CICS termination	Returns to CICS

Table 4. DRA-adapter request summary

Invocation from the DRA	Adapter action
CICS-DBCTL connection is complete	Invoke the control exit
MVS SSI has rejected the IDENTIFY request to DBCTL	Invoke the control exit
DBCTL has rejected the IDENTIFY request	Invoke the control exit

Table 4. DRA-adapter request summary (continued)

Invocation from the DRA	Adapter action
Operator has replied CANCEL to message DFS690A	Invoke the control exit
DBCTL has terminated abnormally	Invoke the control exit
DRA has terminated abnormally	Invoke the control exit
/CHECKPOINT FREEZE has been issued	Invoke the control exit
PSB schedule, DL/I, syncpoint, thread termination, thread purge, or interface termination request is to be suspended	Invoke the suspend exit
PSB schedule, DL/I, syncpoint, thread termination, thread purge, or interface termination request is to be resumed	Invoke the resume exit

Table 5. Adapter exit summary

Circumstances	Adapter action
Successful completion of THREAD SCHED request	Invoke the monitoring exit
Completion of THREAD COMTERM or THREAD ABTTERM request	Invoke the monitoring exit
DRA thread failure	Invoke the status exit
Resynchronization request issued from CICS recovery manager	Invoke the token exit
CICS orderly or immediate term	Invoke the token exit
CICS ABEND	Invoke the token exit
Completion of DRA TERM issued as a result of a termination request from disconnection program	Invoke the statistics exit
Completion of DRA TERM issued as a result of a CICS orderly termination request	Invoke the statistics exit

Suspend exit (DFHDBSPX)

The suspend exit is invoked by the adapter on behalf of the DRA so that a CICS thread can be suspended during the processing of a DL/I call. The suspend exit outputs a trace entry immediately before issuing a WAIT, and a trace entry immediately after it is posted by the resume exit.

The suspend exit is also invoked by the adapter when a disconnection request from the menu is being processed.

Resume exit (DFHDBREX)

The resume exit is invoked asynchronously by the adapter on behalf of the DRA, and it is executed in the DRA environment. It handles both normal resume and abnormal resume after an abend of the thread. The resume exit issues an MVS POST.

When a thread fails, the resume exit is invoked and an ‘unsuccessful’ return code is passed back to the DBCTL call processor, indicating that CICS must issue an abend for that thread (task).

Adapter exits

The following sections describe the adapter exits.

The adapter statistics exit (DFHDBSTX): The statistics exit is invoked by the adapter when the CICS-DBCTL interface has been terminated by the CICS operator using the menu program to request disconnection from DBCTL. The exit is also invoked by the adapter when CICS is terminated in an orderly way.

The function of the exit is to invoke the CICS statistics domain supplying the data that has been returned from the DRA relating to the individual CICS-DBCTL session.

For a /CHECKPOINT FREEZE command, the exit is not invoked, but the statistics domain is called by DFHCDBCT.

The adapter token exit (DFHDBTOX): The token exit is invoked by the adapter when a task is encountered which has not been allocated a thread token, that is, it has not been through the DBCTL call processor module. This occurs for resynchronization tasks and for the CICS termination invocation.

The adapter monitoring exit (DFHDBMOX): The monitoring exit is invoked by the adapter when monitoring data has been returned by DBCTL as a result of a PSB schedule request, and a CICS SYNCPOINT or DLI TERM request. The exit passes the data on to the CICS monitoring domain to update the tasks monitoring information.

The adapter status exit (DFHDBSSX): The status exit is invoked by the adapter in the event of a DRA thread failure, so that resources owned by the failing thread can be transferred to CICS, which then releases the transferred resources during syncpoint processing.

DBCTL system definition

DBCTL system definition is described in the IMS *System Definition Reference*.

DBCTL PSB scheduling

When a CICS task requests the scheduling of a DL/I PSB by means of an EXEC DLI SCHEDULE request or DL/I PCB call, and the request is for a DBCTL PSB, control is passed to DFHDLIDP.

Database calls

For DBCTL, DFHDLIDP invokes the CICS RMI to pass control to DBCTL.

DBCTL PSB termination

DBCTL PSB termination is performed during the syncpoint when the resource manager interface (RMI) communicates with DBCTL.

System termination

Support is provided to close down the CICS-DBCTL interface during CICS termination. This should avoid the possibility of causing DBCTL to terminate with a U113 abend because of CICS terminating while DL/I threads are running on its behalf in DBCTL.

To provide the support, there is an extension to the RMI to invoke active adapters at CICS termination.

If CICS termination hangs because the CICS-DBCTL interface does not close down, the operator should type in a /DISPLAY ACTIVE command on the DBCTL console and identify the threads corresponding to the CICS system being terminated. This is possible because the threads' recovery tokens, which are displayed, start with the CICS applid. The operator should then issue /STOP THREAD requests for each thread.

Control blocks

The following diagram shows the major control blocks used to support the CICS-DBCTL interface:

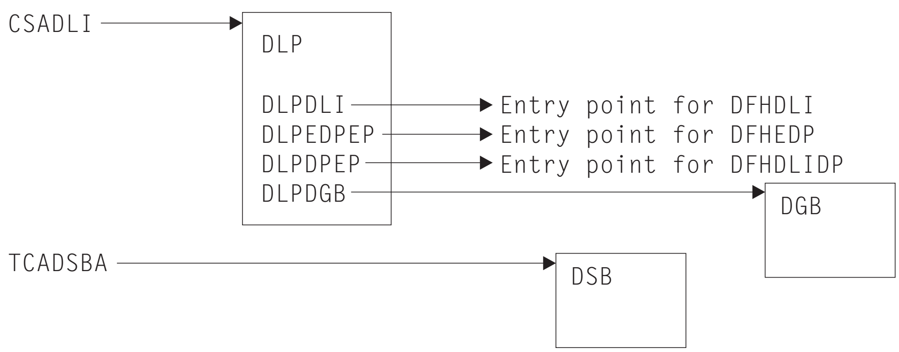


Figure 32. Some control blocks used for DBCTL support

The DL/I interface parameter list (DLP) is described in “DL/I interface parameter list (DLP)” on page 121.

The DBCTL global block (DGB) is acquired, from storage above the 16MB line, when the CICS-DBCTL interface is first initialized. It lasts for the remainder of the CICS execution.

The DBCTL scheduling block (DSB) is acquired, from storage above the 16MB line, when a task issues a PSB schedule request to DBCTL; that is, the PSB used does not appear in the remote PDIR. The DSB is freed at task termination.

See the *CICS Data Areas* manual for a detailed description of these control blocks.

Modules

Module	Description
DFHDBAT	Adapter
DFHDBCON	Initialization program
DFHDBCT	Control program
DFHDBCTX	Control exit
DFHDBDI	Disable program
DFHDBDSC	Termination program
DFHDBIE	Inquiry screens
DFHDBIQ	Inquiry program
DFHDBME	Menu program
DFHDBMOX	Monitoring exit
DFHDBNE	Menu screens
DFHDBREX	Resume exit
DFHDBSPX	Suspend exit
DFHDBSSX	Status exit
DFHDBSTX	Statistics exit
DFHDBTOX	Token exit
DFHDBUEX	DBCTL user exit
DFHDLI	DL/I router program
DFHDLIDP	DBCTL call processor

Exits

The following global user exit points are provided for DBCTL:
- In DFHDBCR: XXDFB and XXDTO
- In DFHDBCT: XXDFA.

For further information about these exit points, see the *CICS Customization Guide* and the *CICS IMS Database Control Guide*.

Chapter 12. Data interchange program

The data interchange program (DFHDIP) supports the batch controller functions of the IBM 3790 Communication System and the IBM 3770 Data Communication System. Support is provided for the transmit, print, message, user, and dump data sets of the 3790 system.

Design overview

The data interchange program is designed as a function manager for Systems Network Architecture (SNA) devices. It is invoked via DFHEDI for command-level requests, or internally by the basic mapping support (BMS) routines using the DFHDI macro. DFHDIP performs the following actions:

1. Determines whether a new output destination has been specified (it retains information about the previous destinations in the data interchange control block) and, if so, builds appropriate FMHs to select the new destination, and outputs these FMHs to the SNA device via terminal control.
2. Invokes the appropriate subroutine to perform the desired function:

   ADD Builds ADD FMH, transmits it and the user data

   REPLACE
   Builds REPLACE FMH, transmits it and the user data

   ERASE
   Builds ERASE FMH and RECID FMH and transmits them

   NOTE Builds NOTE FMH, transmits it, and returns the reply to the user

   QUERY
   Builds QUERY FMH, transmits it, and outputs END FMH

   SEND Outputs user data

   WAIT Waits for completion of the I/O

   END Builds END FMH and transmits it

   ABORT
   Builds ABORT FMH and transmits it

   ATTACH
   Removes FMH from initial input

   DETACH
   Frees the storage used by DFHDIP

   RECEIVE
   Reads a complete record from the logical device.
3. Sets the appropriate return code.

Figure 33 on page 106 shows the data interchange program interfaces.

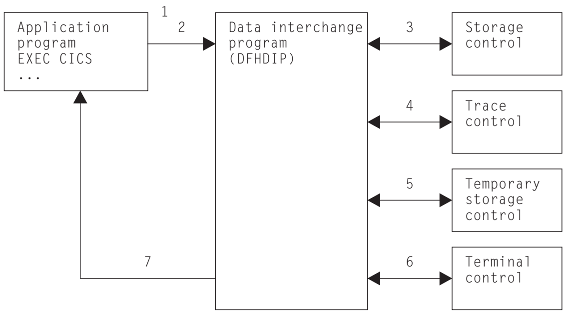


Figure 33. Data interchange program interfaces

Notes:

1. The application program invokes DFHEDI (via DFHEIP) which then communicates with DFHDIP by setting fields in the TCA.
2. DFHDIP receives control.
3. If no storage has been obtained for the data interchange block (DIB), storage control is invoked. The storage is chained to the TCTTE. Significant status information, such as the currently selected destination, is remembered in the data interchange block, which is freed at the end of task processing.
4. A trace entry is made.
5. If logging is present (protected task and message integrity) and if a destination change or function change occurs on output, temporary-storage control is invoked to write the DIB to recoverable temporary storage.
6. Terminal control is invoked to output any built FMH and also to output the user data. (DFHTC TYPE=WRITE is issued.) For input requests, DFHTC TYPE=READ requests are issued to obtain a non-null input record.
7. Any errors obtained from the device are decoded and placed in the TCA return code slot. If no errors were detected, a return code of ‘0’ (zero) is returned.

Modules

DFHEDI, DFHDIP

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for the data interchange program:

- AP 00D7, for which the trace level is DI 1.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 13. Distributed program link

Distributed program link enables a program (the **client program**) in one CICS region to issue an EXEC CICS LINK command to link to a program (the **server program**) running in another CICS region (the **resource region**). The link can be through intermediate CICS regions.

The communication in distributed program link processing is, from the CICS side, synchronous, which means that it occurs during a single invocation of the client program, and that requests and replies between two programs can be directly correlated.

The *CICS Intercommunication Guide* includes information about distributed program link processing.

Figure 34 gives an overview of distributed program link operation.

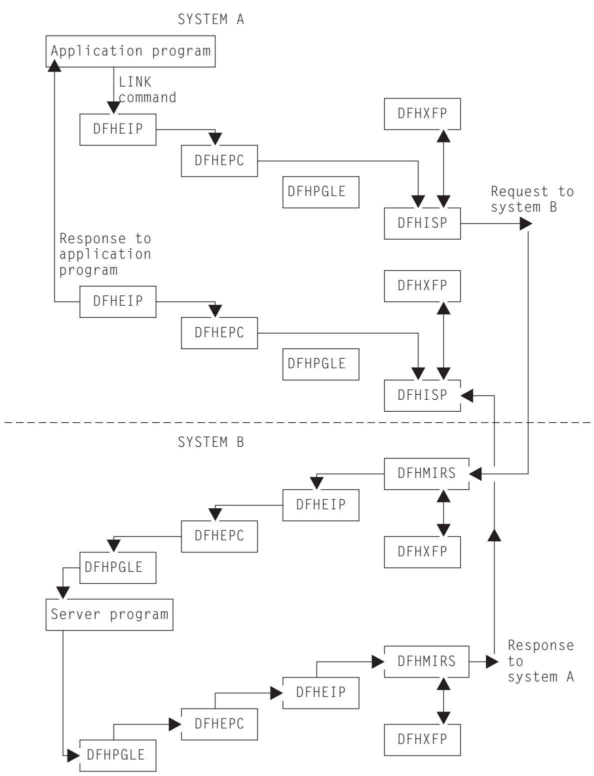


Figure 34. Overview of program link

The DFHEIP module is described in Chapter 19, “EXEC interface,” on page 135. This routes all program control requests to DFHEPC. DFHEPC passes all remote LINK requests to the program manager domain (PGLE_LINK_EXEC request). For local programs, program manager links to the program and, on return, it returns to DFHEPC. For remote programs, program manager returns to DFHEPC with and exception response, with a reason code indicating “remote program”, and DFHEPC passes the request to the intersystems program, DFHISP. The operation of DFHISP for distributed program link is the same as for function shipping, but only the DFHXFP transformations are used. (See Chapter 26, “Function shipping,” on page 277.) The operation of DFHPEP is described in Chapter 38, “Program control,” on page 333; the interface to DFHPGLE LINK_EXEC is described in Chapter 95, “Program manager domain (PG),” on page 949.

CICS handles session failures and systems failures for distributed program link processing by returning a TERMERR condition to the program that issued the LINK request.

If the server program terminates abnormally and does not handle the abend itself, DFHMIRS returns the abend code to the program that issued the LINK request. This code is the last abend code to occur in the server program, which may have handled other abends before terminating.

A client program using distributed program link can specify that a SYNCPOINT is to be taken in the resource region on successful completion of the server program. That is, any resources updated by the server program (or any associated program) are treated as if they are a separate unit of work.

Modules

The following modules are involved in the distributed program link:

DFHEIP
: EXEC interface (see Chapter 19, "EXEC interface," on page 135)

DFHEPC
: DFHEIP program control interface (see Chapter 38, "Program control," on page 333)

DFHISP
: ISC converse (see Chapter 26, "Function shipping," on page 277)

DFHMIRS
: Mirror transaction (see Chapter 26, "Function shipping," on page 277)

DFHPGLE
: PG domain - link exec function (see Chapter 95, "Program manager domain (PG)," on page 949)

DFHXFP
: Online data transformation program (see page 1489)

Exits

There are two global user exit points in DFHEPC: XPCREQ and XPCREQC.

Trace

No trace points are provided for this function.

Chapter 14. Distributed transaction processing

Distributed transaction processing enables a CICS transaction to communicate with a transaction running in another system. The transactions are designed and coded explicitly to communicate with each other, and thereby to use the intersystem link with maximum efficiency.

The communication in distributed transaction processing is, from the CICS side, synchronous, which means that it occurs during a single invocation of the CICS transaction and that requests and replies between two transactions can be directly correlated.

The *CICS Intercommunication Guide* tells you about multiregion operation and intersystem communication, and also includes some information about distributed transaction processing. Guidance information about designing and developing distributed applications is given in the *CICS Distributed Transaction Programming Guide*.

Design overview

CICS handles session failures and systems failures for distributed transaction processing in the same way as for CICS function shipping. See the relevant sections in Chapter 26, “Function shipping,” on page 277 for further information.

Distributed transaction processing with MRO and LU6.1

Figure 35 gives an overview of the modules involved with distributed transaction processing for MRO and LU6.1 ISC.

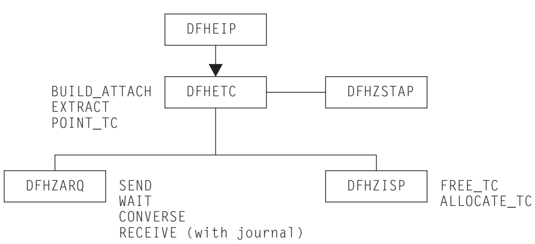


Figure 35. Distributed transaction processing for MRO and LU6.1

The DFHEIP module is described in Chapter 19, “EXEC interface,” on page 135. This routes all terminal control requests to DFHETC. DFHETC handles BUILD_ATTACH, EXTRACT, and POINT_TC requests itself. It routes all other requests (SEND, WAIT, CONVERSE, RECEIVE (with journal)), to DFHZARQ, except for FREE_TC and ALLOCATE_TC requests, which are routed to DFHZISP. If the request requires that the user conversation state be returned, DFHETC calls DFHZSTAP. All these modules are described in detail under “Modules” on page 111.

Mapped and unmapped conversations (LU6.2)

In **mapped** conversations, the data passed to and received from the LU6.2 application programming interface (API) is simply user data. Mapped conversations use the normal CICS API. Application programs and function shipping requests written for LU6.1 operate using mapped conversations when transferred to LU6.2.

Figure 36 on page 110 gives an overview of the modules involved with the processing of mapped conversations in LU6.2. ISC.

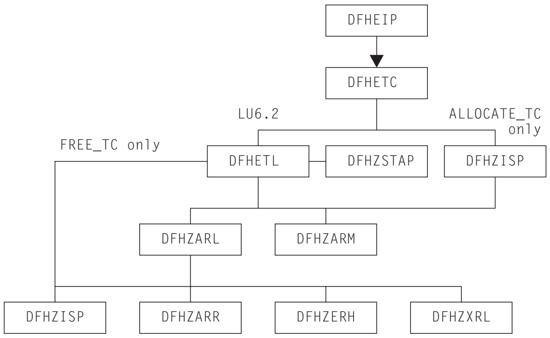


Figure 36. Distributed transaction processing for mapped conversations in LU6.2

The DFHEIP module is described in Chapter 19, “EXEC interface,” on page 135. This routes all terminal control requests to DFHETC. DFHETC routes all requests relating to an LU6.2 session to DFHETL except for ALLOCATE_TC requests, which are routed to DFHZISP.

In turn, DFHETL calls DFHZARL to process most requests; it calls DFHZISP to handle FREE_TC requests, and DFHZARM to handle the receipt of unrecognized or unsupported IDs. If the request requires that the user conversation state be returned, DFHETL calls DFHZSTAP.

DFHZARL’s processing depends on the type of request; for example, it calls DFHZISP to allocate a TCTTE, DFHZARR to receive data, and DFHZERH for outbound or inbound FMH7 processing. If the request needs to be transaction routed, DFHZARL calls DFHZXRL to route the request to the terminal-owning region (see Chapter 62, “Transaction routing,” on page 441).

With the exception of DFHZXRL, all these modules are described in detail under “Modules” on page 111.

Unmapped conversations (also known as **basic** conversations), are used principally for communication with device-level products that do not support mapped conversations, and which possibly do not have an API open to the user. In unmapped conversations, the data passed to and received from the LU6.2 API contains GDS headers.

Figure 37 gives an overview of the modules involved with the processing of unmapped conversations in LU6.2 ISC.

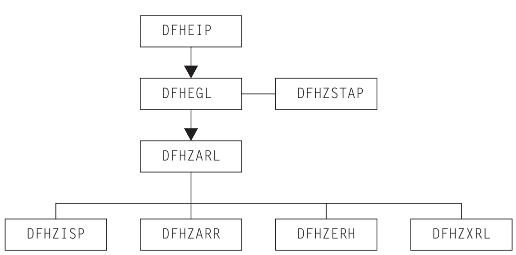


Figure 37. Distributed transaction processing for unmapped conversations in LU6.2

The DFHEIP module is described in Chapter 19, “EXEC interface,” on page 135. This passes control to DFHEGL to process GDS commands. DFHEGL routes all GDS conversation-related commands directly to DFHZARL. Some validation of application-provided parameters is performed, and errors are reflected back to the application. If the request requires that the user conversation state be returned, DFHEGL calls DFHZSTAP.

DFHZARL's processing depends on the type of request; for example, it calls DFHZISP to allocate a TCTTE, DFHZARR to receive data, and DFHZERH for outbound or inbound FMH7 processing. If the request needs to be transaction routed, DFHZARL calls DFHZXRL to route the request to the terminal-owning region (see Chapter 62, "Transaction routing," on page 441).

With the exception of DFHZXRL, all these modules are described in detail in the next section.

Modules

DFHEGL

DFHEGL processes GDS commands. It is an EXEC interface processor module, and receives control directly from DFHEIP. The TCTTE for the session is located and checked for validity. All GDS conversation-related commands are mapped into a DFHLUC macro call and routed directly to DFHZARL. There is no mapping or unmapping of data, state indicators are not maintained, and there are no FMHs to process.

DFHETC and DFHETL

DFHEIP routes all terminal control requests to DFHETC (the EXEC interface processor for terminal control). DFHETC handles BUILD_ATTACH, EXTRACT, and POINT_TC requests itself. It routes all other requests relating to an MRO or LU6.1 session to DFHZARQ except for FREE_TC and ALLOCATE_TC requests, which are routed to DFHZISP. It routes all other requests relating to an LU6.2 session to DFHETL except for ALLOCATE_TC, which is routed to DFHZISP.

DFHETL performs the following actions:

1. Maps an application request into a form suitable for the DFHZCP and DFHZCC application request modules. This includes mapping application data into GDS records.
2. Detects errors and returns error codes to the application.
3. Unmaps data from GDS records.
4. Maintains state indicators.

For ISSUE CONFIRMATION, CONNECT PROCESS, EXTRACT PROCESS, ISSUE ERROR, ISSUE ABEND, and ISSUE SIGNAL commands, DFHETL:

1. Maps application requests into DFHLUC macro calls.
2. Updates state indicators in the TCTTE (for example, the TCTTE indicator that shows that a CONNECT PROCESS command has been issued).

For SEND and CONVERSE commands, DFHETL:

1. Obtains storage for the processing of outbound application data.
2. Creates attach FMHs, if appropriate.
3. Calls DFHZARL to transmit data.

For RECEIVE commands, DFHETL:

1. Obtains storage for the processing of inbound data.
2. Calls DFHZARL to receive inbound data.
3. Extracts inbound FMHs, as appropriate.
4. Unmaps inbound data.
5. Validates LLs and rejects them if not valid.
6. Manages the passing of data back to the application.

7. If the application issues a RECEIVE NOTRUNCATE request in order to receive only part of the chain, retains the residual data for subsequent RECEIVE requests. DFHETL receives one complete chain of data at a time from DFHZARL.

For WAIT commands, DFHETL calls DFHZARL.

For FREE commands, DFHETL:
1. Checks that the terminal is in the correct state to be freed.
2. Frees the storage used to hold RECEIVE data and the ETCB.
3. Calls DFHZISP to free the session.

DFHZARL

DFHZARL is always invoked via the DFHLUC macro. The DFHLUCDS DSECT maps a parameter list that is set up to pass information to and return information from DFHZARL. DFHZARL manages data in buffers, not in TIOAs. SEND commands cause data to be assembled by DFHZARL into a buffer until a WAIT, or other event, causes the data in the buffer to be transmitted.

DFHZARL invokes DFHZSDL to send data to VTAM, by placing requests on the activate chain. However, for optimization, DFHZARL can invoke DFHZSDL directly. Receive requests are handled by DFHZARR.

DFHZARL invokes DFHZUSR to manage the conversation state. The LU6.2 states for each session are stored in the TCTTE for that session.

If the request needs to be transaction routed, DFHZARL calls DFHZXRL to route the request to the terminal-owning region (see Chapter 62, “Transaction routing,” on page 441).

Details of DFHZARL’s processing for the principal functions of the DFHLUC macro that is used to invoke DFHZARL are given below.

INITIAL_CALL function

This function is requested by DFHZSUP. DFHZARL acquires LU6.2 send and receive buffers. If the transaction is being started as a result of an ATTACH request received from a remote system, DFHZARL transfers any data received with the attach header from the TIOA into the receive buffer.

ALLOCATE function

DFHZARL performs the following actions:
1. If the request passed the address of a profile entry, puts this address in the TCA. If the request passed the name of a profile, calls transaction manager to locate the entry and then puts the address of the entry in the TCA.
2. If the request passed a netname rather than a specific sysid, calls DFHZLOC to locate the TCTTE for the netname and then puts the sysid into the DFHLUC parameter list (as if the caller had the specified sysid).
3. Copies the DFHLUC parameter list to LIFO storage.
4. Calls DFHZISP to allocate a TCTTE.
5. Addresses the TCTTE allocated.
6. Acquires LU6.2 send and receive buffers.
7. Sets the user state machine (DFHZUSRM), request = ALLOCATE_RESOURCE.
8. Returns results to the caller.

SEND function

DFHZARL performs the following actions:
1. Checks the user state machine (DFHZUSRM).

2. Checks the LL count and maintains a record of the outstanding LL count.
3. If the command is SEND LAST, INVITE, or CONFIRM, and the outstanding LL count is nonzero, issues an error message.
4. Sets the user state machine (DFHZUSRM).
5. Issues RECEIVE IMMEDIATE requests, as required, to pick up any negative responses sent by the partner program.

The caller must specify WAIT in the request to force the data to be sent immediately. SEND CONFIRM has an implicit WAIT, and control is not returned until a response has been received, when the state machine is set.

For a SEND request with WAIT, DFHZARL then:
1. Sets the user state machine (DFHZUSRM), request=WAIT.
2. Invokes DFHZSDL for transmission of the data in application area or send buffer.

For a SEND request without WAIT, DFHZARL then:
1. If there is sufficient space in the send buffer for all the data, transfers the data from the application area to the send buffer, and returns control to the caller.
2. Saves the INVITE and LAST indicators.
3. If the send buffer cannot hold all the data, invokes DFHZSDL for an implicit SEND.

If data or a CONFIRM command was sent (or both), DFHZARL then:
1. Checks for a signal received.
2. Checks for exception (negative) response received. If found, calls DFHZERH to handle the error. On return, sets the state machine.
3. Returns results to the caller.

When an implicit send is required, DFHZARL passes the data to DFHZSDL for transmission, passing the address of the data in the send buffer and in the application buffer. The total length of data passed to DFHZSDL is a multiple of the request unit size. On return to DFHZARL, the remaining data is transferred to the send buffer. The parameters passed to DFHZARL, such as INVITE and LAST, are not transmitted by DFHZSDL.

RECEIVE function

DFHZARL passes the DFHLUC parameter list, specifying the type of receive required, to DFHZARR for processing (see “DFHZARR” on page 115).

ISSUE ERROR or ABEND function

DFHZARL is called as a result of an ISSUE ERROR or ISSUE ABEND command, and performs the following actions:
1. Sets the user state machine
2. Calls DFHZERH.

DFHZARM

DFHETL may invoke DFHZARM to provide service functions. DFHZARQ passes control to DFHZARM instead of initiating DFHZSDS, DFHZRVS, and so on, if DFHZARQ finds that it is an LU6.2 session. This applies to the SEND, WAIT, RECEIVE, and SIGNAL commands. The same applies to DFHZISP for the FREE command.

DFHZARM translates the data stream to and from a format suitable for invoking DFHZARL. In particular:
- An LU6.2 attach FMH may have to be requested.
- Data must be passed in GDS record format (structured fields preceded by an LLID).

DFHZARM is invoked via the DFHLUCM macro, which has seven executable options:

DFHLUCM TYPE =
- SEND
- RECEIVE
- WAIT
- SIGNAL
- FREE
- INVALID_ID

DFHLUCM TYPE=STORAGE defines the storage in LIFO for passing primary input and output. The DSECT name is DFHLUMDS. TCTTE contains the secondary input and output. The principal functions are described in the following sections.

SEND function

DFHZARM performs the following actions:

1. Maps the data into GDS record format. The IDs used are:
   - X'12F1'
   - X'12F2'
   - X'12FF'.
2. Examines bits set in the TCTTE by DFHZARL to determine which DFC to apply.
3. Invokes DFHZARL (using a DFHLUC TYPE=SEND,LIST=... macro call) to pass the GDS records and DFC indicators.
4. Updates the state bits in TCTTE as necessary.
5. Interrogates the LU6.2 ATTACH_FMH_BUILT bit in the TCTTE, which was set by DFHZSUP or DFHETL. This bit indicates whether this is first SEND. If an LU6.2 attach header has not already been built as a result of a CONNECT PROCESS command, DFHZARM issues CONNECT_PROCESS to DFHZARL, assuming synclevel 2, before sending the data.

RECEIVE function

DFHZARM performs the following actions:

1. Calls DFHZARL using TYPE=BUFFER. Two calls are made. On the first call, the first 4 bytes (LLID) are retrieved into LIFO. These are examined and the LL is used to determine the TIOA size and to specify the length required in the second call.
2. On the second call, retrieves the remainder of the data directly into the TIOA. If the LL indicates concatenated data, a series of calls is made to retrieve all the data.

FREE function

The FREE function is used, for example, by DFHZISP to ensure that I/O has completed and CEB sent, using null data if necessary.

INVALID_ID function

The INVALID_ID function is used by DFHETL and DFHZARM itself. It handles the receipt of unrecognized or unsupported IDs. DFHZARM calls DFHZARL with ISSUE_ERROR (X'0889010x'), and sends a record with ID X'12F4' followed by the unrecognized ID. If the remote system responds, DFHZARM turns the flows around so that the local system can try again.

LU6.1 chains

An LU6.1 chain corresponds to one SEND command. LU6.2 chains are bigger, so:

- For outbound data, DFHZARM maps one SEND into one structured field (concatenated if necessary).
- For inbound data, DFHZARM retrieves one (possibly concatenated) field and calls it a chain, thus preserving compatibility.

DFHZARQ

DFHETC routes SEND, WAIT, CONVERSE, and some RECEIVE commands to DFHZARQ. RECEIVE commands are passed to DFHZARQ if input journaling is in effect. Otherwise, the call is routed to DFHZARL directly.

DFHZARQ passes control to DFHZARM instead of initiating DFHZSDS, DFHZRVS, and so on, if DFHZARQ finds that it is an LU6.2 session. This applies to the SEND, WAIT, RECEIVE, and SIGNAL commands.

Reasons for calling DFHZARQ are:

- To avoid duplication of existing code
- So that DFHZCP performs journaling of outbound data
- To perform an implicit CONNECT PROCESS if SEND or CONVERSE is the next session-related command after ALLOCATE
- To enable the SNA change direction (CD) and end bracket (EB) indicators to flow with the data.

DFHZARR

DFHZARR is called by DFHZARL to handle receive requests. Details of the processing follow.

RECEIVE function

This function must be able to handle receipt of the following:

- Application data
- FMH7s and ER1s (negative responses)
- PS_headers (Prepares, Request_commits)
- Indicators such as CD, CEB, and RQD2
- Signal.

Figure 38 gives an overview of the modules involved with the processing of receive requests. These modules are described in Chapter 116, “CICS executable modules,” on page 1449.

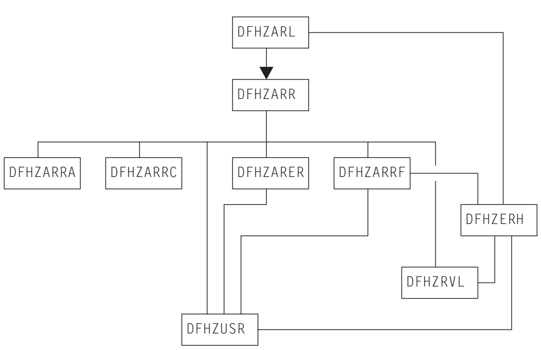


Figure 38. Distributed transaction processing of LU6.2 receive requests

DFHZARL passes the DFHLUC parameter list, specifying the type of receive required, to DFHZARR.

DFHZARR then performs the following actions:

1. Checks that request is valid; if not, returns error codes.
2. Initializes the application and LU6.2 receive buffers (by calls to DFHZARRA and the DFHZARR0 subroutine of DFHZARR respectively).
3. Calls DFHZARRC to determine what to process next.
4. Depending on DFHZARRC’s response, calls the relevant subroutine.
5. If “enough” (or all that can be) has not been received, loops back to step 3; otherwise step 6.

6. Tests for (and returns) signal when it has been received.

The results of the receive are passed back to the caller in the DFHLUC parameter list.

To control this processing, DFHZARR uses the variables **receive_type** and **what_next**, as follows.

receive_type can have the following values:

RECEIVE_WAIT
Request was a receive and wait.

RECEIVE_IMMEDIATE
Request was a receive immediate.

LOOK_AHEAD
All the allowed user data has been received, but only one receive immediate call to the DFHZARR1 subroutine of DFHZARR is permitted to attempt to pick up indicators such as CD, CEB, or a PS_header.

NO_MORE_RECEIVES
No more calls to DFHZARR1 are permitted, but processing may continue with what has already been received.

NO_RECEIVE_LOOK_AHEAD
All the allowed user data has been received. An attempt must be made to pick up indicators such as CD, CEB, or a PS_header without a call to DFHZARR1. This value is only required for a receive immediate request.

RECEIVE_COMPLETE
Receive processing is finished.

The first two values are possible initial values of receive_type, and the other four are used as the receive progresses.

what_next is an output of DFHZARRC, and represents what is next to be processed. It can have the following values:

DATA_RECORD
Application data

FMH_RECORD
FMH7 in the buffer

PS_HEADER_RECORD
Prepare or Request_commit

PARTIAL_LL
First byte of a logical record only, therefore cannot tell whether it is a DATA_RECORD or PS_HEADER_RECORD

CD
Change Direction

CEB
Conditional End Bracket

RQD2
RQD2 without CD or CEB

RQD2_CD
RQD2 with CD

RQD2_CEB
RQD2 with CEB

ER1
Negative response

EMPTY_BUFFER
Nothing available to receive.

DFHZERH

DFHZERH is called by DFHZARL or DFHZARRF, when it is required to transmit error information or when error information has been received.

Outbound errors

For outbound errors, DFHZERH is invoked by DFHZARL following an ISSUE_ERROR, ISSUE_ABEND, or SYNC_ROLLBACK request.

An FMH7 must be transmitted, but can only be transmitted if the session is in the send state.

If the session is in the receive state, DFHZERH:
1. Sends a negative response
2. Purges the remaining data to end of chain.

In all cases, DFHZERH then:
1. Checks that the session is still in bracket
2. Clears the send buffers
3. Calls DFHZARL to send the FMH7.

Inbound errors

For inbound errors, DFHZERH is invoked by DFHZARL or DFHZARRF when a process-level exception response or an FMH7 has been received.

If an exception response is received while in the send state, DFHZERH purges the present output buffer and sends ‘LIC,CD,RQE1’ to put the conversation into receive state—so that the following FMH7 can be received.

If an FMH7 is received, DFHZERH examines the associated sense code and any GDS error log data, then returns to its caller.

DFHZISP

DFHZISP is called by DFHETC to perform ALLOCATE_TC requests. (ALLOCATE commands are passed to DFHZISP because DFHETC cannot check the session type until the session is allocated.)

DFHZISP is also called to perform FREE_TC requests.

DFHZSTAP

DFHZSTAP provides a means of determining the conversation state of an MRO or LU6.2 session from the application side. This function is required if the application issues an EXEC CICS EXTRACT ATTRIBUTES command with the STATE option, or a conversation-based command with the STATE option.

For MRO, modules that invoke MVS services via the DFHTC macro also update the conversation state information with a DFHZCNVM TYPE=PUT macro call. When an application requires the conversation state of a session, DFHETC calls DFHZSTAP using a DFHZSTAM TYPE=GETCURRSTATE macro, which returns a value representing the conversation state of the session.

For LU6.2, DFHZUSR is called to maintain the user conversation state machine. (See Chapter 66, “VTAM LU6.2,” on page 479 for further details.) When an application requires the conversation state of a session, DFHETL (mapped) or DFHEGL (unmapped) calls DFHZSTAP using a DFHZSTAM TYPE=GETCURRSTATE macro. DFHZSTAP examines the DFHZUSR state machine and maps the information into a value representing the conversation state of the session.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for distributed transaction processing:

- AP FDxx, for which the trace level is TC 1
- AP FExx (LU6.2 application receive requests), for which the trace levels are TC 2 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 15. DL/I database support

Facilities for accessing DL/I databases and database control (DBCTL) support are available only with IMS.

Within a single CICS system, the following types of support can be available:

- DBCTL support present. For specific information about DBCTL, see Chapter 11, "Database control (DBCTL)," on page 93.
- Remote DL/I and DBCTL support present (the PDIR system initialization parameter is specified). For specific information about remote DL/I, see Chapter 41, "Remote DL/I," on page 341.

The rest of this section covers DL/I database support in general.

Design overview

The following types of DL/I requests can be made by a CICS system:

- EXEC DLI statements (converted into standard CALL DLI statements by DFHEDP)
- CALL DLI statements.

CICS support for DL/I is provided as follows:

1. A router component

   This component determines whether the call is using a remote or DBCTL PSB, and passes control to the appropriate call processor. This component is described in more detail later in this section.
2. A DL/I call processor

   This component is subdivided into:
   - A remote DL/I call processor
   - A DBCTL DL/I call processor.

   Each call processor deals with a specific interface that is described in the appropriate section of this book for the remote DL/I function and the DBCTL function.

Figure 39 shows the relationships between the components of the CICS-DL/I interface.

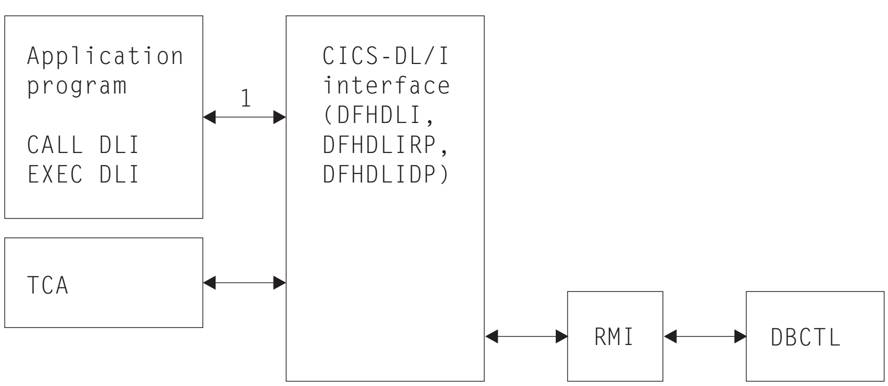


Figure 39. CICS-DL/I interfaces

Notes:

1. When DL/I functions are requested by an application program or a CICS control module through execution of a CALL or CALLDLI macro, DFHDLI sets the required fields in the TCA. EXEC DLI statements are converted into standard CALL DLI statements by DFHEDP.

If the request is for a remote database, DFHDLI passes control to DFHDLIRP. If the request is for a DBCTL database, DFHDLI passes control to DFHDLIDP.

In addition to processing DL/I input/output requests, the DL/I interface, on request, schedules and terminates DL/I program specification blocks (PSBs).

The remainder of this section is concerned with the router component.

The router component (DFHDLI)

The router component receives a request in standard CALL DLI parameter lists. At schedule time, it determines whether the request is a remote or DBCTL request.

Amongst the functions of the router are the following:

Deciding where to process a request

At PSB schedule time, the router determines whether the DL/I requests issued from the application program should be routed to DBCTL or another CICS system (remote). The presence (or absence) of the PSB used in the PDIR determines where the call gets routed.

If no PDIR exists (that is, the PDIR=NO system initialization parameter is specified or is allowed to default), the request is routed to the DBCTL call processor.

If a PDIR has been specified, the router module scans the PDIR. All entries in the PDIR have a SYSIDNT option specified. If the PSB is not found in the PDIR, or if the PDIR entry specifies a SYSIDNT that is the SYSIDNT of the CICS system that is currently running, the request is routed to the DBCTL call processor. Otherwise, the request is routed to the remote call processor.

All DL/I requests are routed to the same DL/I call processor as the corresponding PSB schedule request in the same unit of work.

Initiating synchronization processing

The router provides special handling of the DL/I TERM call. When the router detects a TERM call, it forces a syncpoint, causing CICS to carry out syncpoint processing for the task.

Generating CICS trace records

The router module generates CICS trace records at DL/I call entry and DL/I call exit.

Control blocks

DL/I database support uses the control blocks DIB, DLP, and UIB, which are shown in Figure 40 on page 121.

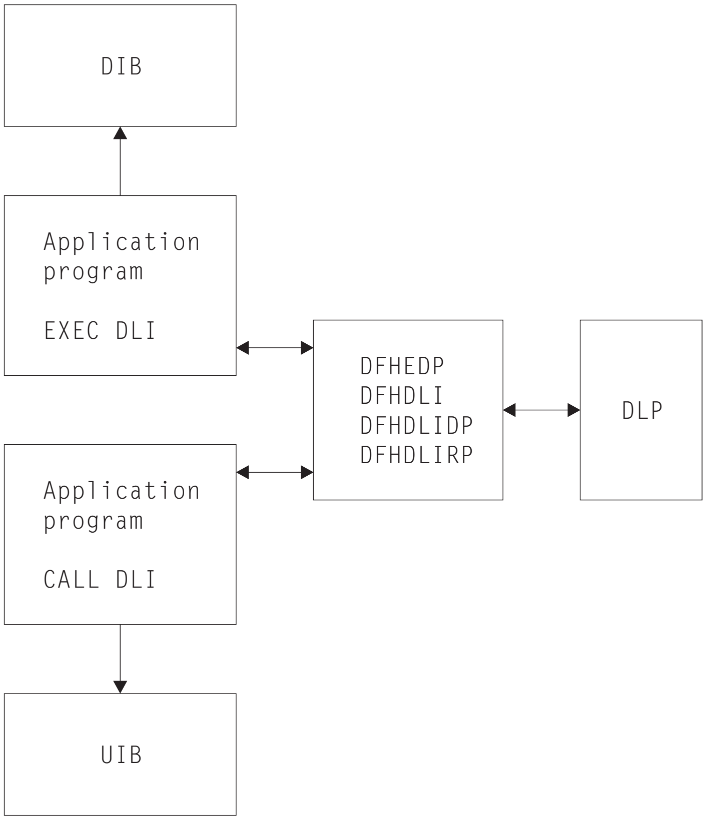


Figure 40. Control blocks for DL/I database support

DL/I interface block (DIB)

When an application program issues EXEC DLI requests, it uses the user DL/I interface block (DIB) instead of the user interface block (UIB). On return, DFHEDP extracts data from the UIB to place in the DIB. The storage for the user DIB is part of the application program. The definition of the user DIB is automatically inserted by the CICS translator for an EXEC DLI application program.

DL/I interface parameter list (DLP)

The DL/I interface parameter list (DLP) is a global DL/I interface control block that lasts for the duration of a CICS session, and contains information relating to the type of DL/I support present in the CICS system. The DLP is created during CICS startup and is addressed by CSADLI in the CSA optional features list.

See *CICS Data Areas* for a detailed description of this control blocks.

User interface block (UIB)

The user interface block (UIB) is the control block used by the CALL and CALL DL/I interfaces to pass response codes and the PCB address list to application programs using CALL DL/I services. The UIB is acquired when a task issues its first PSB schedule request specifying that it requires a UIB. The UIB is freed at task termination. TCADLIBA points to the UIB.

See the *CICS Data Areas* manual for a detailed description of these control blocks.

Modules

Figure 41 shows the module flow of DL/I requests to the DL/I call processors. DL/I requests from application programs made using CALL or CALL DL/I are handled by DFHEIP. Requests made using EXEC DLI are passed from DFHEIP, to the RMI, to DFHEDP. Next, three main CICS-DL/I interface modules process the requests. The first module, DFHDLI, determines what sort of DL/I request is being made and then passes control to one of two call processors. These are the DBCTL DL/I call processor, DFHDLIDP, and the remote call processor, DFHDLIRP. DFHDLIDP routes the requests to the RMI, then DFHDBAT, to IMS/ESA® modules. DFHDLIRP routes the request to DFHISP.

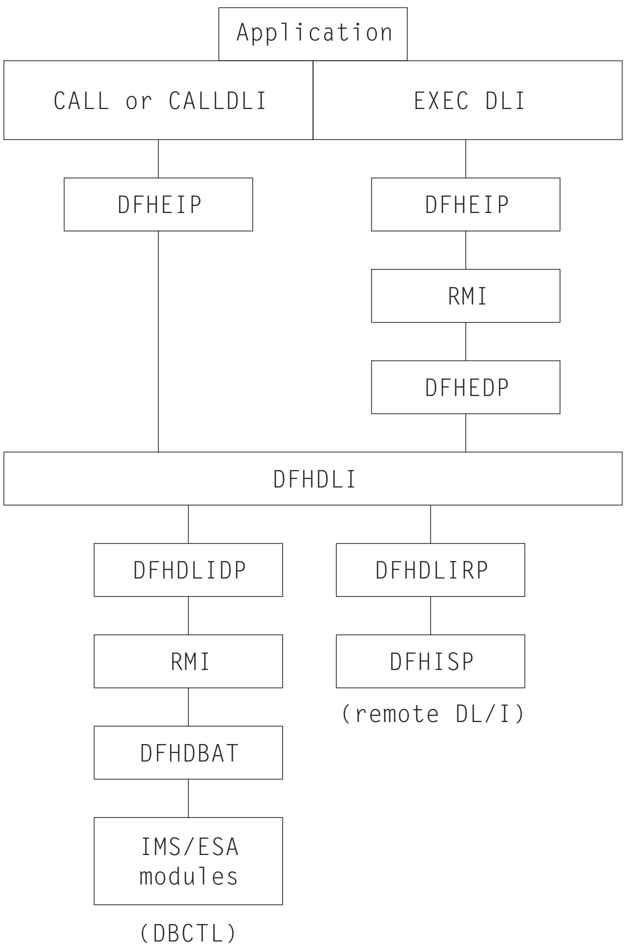


Figure 41. Module flow of DL/I requests to the DL/I call processors

The common CICS-DL/I interface modules consist of the following:

- DFHDLI—contains the code for routing requests to DFHDLIRP and DFHDLIDP
- DFHDLIDP—contains the code for DBCTL requests.
- DFHDLIRP—contains the code for remote DL/I requests

Exits

The following global user exit points are provided in DFHDLI: XDLIPRE and XDLIPOST. For further information about these, see the *CICS Customization Guide* and the *CICS IMS Database Control Guide*.

Trace

The following point ID is provided for DL/I and DBCTL:
- AP 03xx, for which the trace levels are FC 1, FC 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 16. Dump utility program (DFHDU640)

The dump utility program (DFHDU640) runs offline (in batch mode) to produce a printout of the CICS transaction dumps from a CICS transaction dump data set (DFHDMPA or DFHDMPB).

Design overview

DFHDU640 operates in batch mode while one of the dump data sets is closed. Each area, program, and table entry is identified, formatted, and printed separately, with both actual and relative addresses to facilitate analysis. You can select single or double spacing of dumps when the dump utility program is executed.

The CICS dump data set (DFHDMPA or DFHDMPB) contains a number of CICS transaction dumps. These are produced as the result of a transaction abend or a user-application EXEC CICS DUMP TRANSACTION request.

DFHDU640 runs as a stand-alone program in batch mode to format and print the contents of a transaction dump data set. Parameters specified on the SYSIN data set can be used to print only selected dumps or an index of the dumps in the data set.

For further details about DFHDU640, see the *CICS Operations and Utilities Guide*.

Data sets

There are three sources of data for DFHDU640:

Parameters on JCL EXEC statement
A character string of keywords that can be specified to control the layout and format of the dumps.

SYSIN
Records specifying the criteria to be used in selecting which of the dumps on the data set are to be printed.

DFHDMPDS
The transaction dump data set.

There are two output files:

DFHPRINT
The print file for the formatted transaction dump.

DFHTINDX
The print file for the index of dumps on the data set.

Processing

Figure 42 shows the dump utility program interfaces.

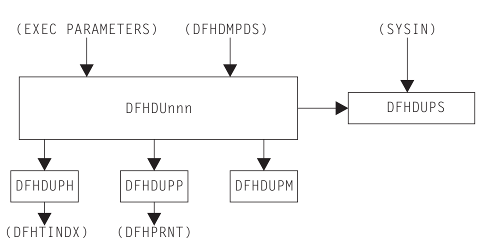


Figure 42. Dump utility program interfaces

The overall flow of the processing within DFHDU640 is as follows. Unless otherwise indicated, all processing is performed by DFHDUPR, the main component of DFHDU640.

1. Process the EXEC parameters if they are present.
2. Call DFHDUPP to open the print data set DFHPRINT.

3. Open the dump data set DFHDMPDS.
4. Read the dumps from DFHDMPDS. For each dump there are four categories of records:

   Dump header record
   Call DFHDUPS to see whether this dump is required for printing. On the first time through, DFHDUPS reads the selective print information from SYSIN. DFHDUPS also calls DFHDUPH to add the dump to the dump index data set DFHTINDX. DFHDUPH opens DFHTINDX on its first invocation.

   Module index records
   DFHDUPM is called to accumulate the module index information in a table in main storage.

   Other data records
   The data is formatted into print lines and DFHDUPP is invoked to write them to DFHPRINT.

   Dump trailer record
   DFHDUPM is invoked to sort and format the module index records. DFHDUPP is called to write them to DFHPRINT.

5. When the end of the dump data set is encountered:
   a. DFHDUPP is called to close DFHPRINT.
   b. DFHDUPH is called to close DFHTINDX.
   c. DFHDUPR closes DFHDMPDS.
6. DFHDU640 terminates.

Modules

Module	Function
DFHDUPR	Controlling routine, responsible for reading information from the dump data set DFHDMPDS.
DFHDUPS	Receives the address of a dump header record from the dump data set, and decides whether this dump fulfils the criteria for printing. On first entry, reads and stores the selective print parameters from SYSIN.
DFHDUPP	Is responsible for all access to the print file DFHPRINT, namely for OPEN, CLOSE, and PUT requests.
DFHDUPH	Writes line to dump index for each dump header record encountered. On first entry, opens the index file DFHTINDX.
DFHDUPM	Invoked for each module index entry found to save information. Invoked when dump trailer record found to format and print the complete module index.

Copy books

Copy book	Function
DFHDUPSC	Contains the definition of the parameter list passed to DFHDUPS.
DFHDUPMC	Contains the definition of the parameter list passed to DFHDUPM.
DFHDUPPC	Contains the definition of the parameter list passed to DFHDUPP.

Exits

Global user exit points are not applicable to offline utilities.

Trace

Trace points are not applicable to offline utilities.

Chapter 17. Dynamic allocation sample program (IBM 3270 only)

Any data set defined to file control can be allocated to CICS dynamically when the file is opened, rather than at CICS job initiation time. This allocation takes place automatically if job control statements for the data set are not included in the CICS job stream, and if both the data set name and the disposition have been specified in the file control table when the data set is opened.

The dynamic allocation sample program provides an alternative way to perform dynamic allocation. When used with a terminal of the IBM 3270 Information Display System, it gives the user access to the functions of DYNALLOC (SVC 99) in MVS. This can be used, in conjunction with master terminal functions and suitable operating procedures, to allocate and deallocate any file that CICS can dynamically open and close.

Design overview

The program runs as a CICS transaction, using CICS functions at the command level wherever possible. It does not modify any CICS control blocks. Only the DYNALLOC function is available through the program; any manipulation of the environment before or after the DYNALLOC request must be done by other means.

CICS supplies sample resource definitions for the program load module, DFH99, and the transaction, ADYN, that invokes it. These definitions are in the group DFH$UTIL. Note that DFH99 ***must*** be defined with EXECKEY(CICS).

The flow in a normal invocation is as follows. The main program, DFH99M, receives control from CICS, and carries out initialization. This includes determining the screen size and allocating input and output buffer sections, and issuing initial messages. It then invokes DFH99GI to get the input command from the terminal. Upon return, if the command was null, the main program terminates, issuing a final message.

The command obtained has its start and end addresses stored in the global communication area, COMM. The main program allocates storage for tokenized text, and calls DFH99TK to tokenize the command. If errors were detected at this stage, further analysis of the command is bypassed.

Following successful tokenizing, the main program calls DFH99FP to analyze the verb keyword. DFH99FP calls DFH99LK to look up the verb keyword in the table, DFH99T. DFH99LK calls DFH99MT if an abbreviation is possible. Upon finding the matching verb, DFH99FP puts the address of the operand section of the table into COMM, and puts the function code into the DYNALLOC request block.

The main program now calls DFH99KO to process the operand keywords. Each keyword in turn is looked up in the table by calling DFH99LK, and the value coded for the keyword is checked against the attributes in the table. DFH99KO then starts off a text unit with the appropriate code, and, depending on the attributes the value should have, calls a conversion routine

- For character and numeric strings, DFH99CC is called. It validates the string, and puts its length and value into the text unit.
- For binary variables, DFH99BC is called. It validates the value, converts it to binary of the required length, and puts its length and value into the text unit.
- For keyword values, DFH99KC is called. It looks up the value in the description part of the keyword table using DFH99LK, and puts the coded equivalent value and its length into the text unit.

When a keyword specifying a returned value is encountered, DFH99KO makes an entry on the returned value chain, which is anchored in COMM. This addresses the keyword entry in DFH99T, the text unit where the value is returned, and the next entry. In this case the conversion routine is still called, but it only reserves storage in the text unit, setting the length to the maximum and the value to zeros.

When all the operand keywords have been processed, DFH99KO returns to the main program, which calls DFH99DY to issue the DYNALLOC request.

DFH99DY sets up the remaining parts of the parameter list, and if no errors too severe have been detected, a subtask is attached to issue the DYNALLOC SVC. A WAIT EVENT is then issued against the subtask termination ECB. When the subtask ends, and CICS dispatches the program again, the DYNALLOC return code is captured from the subtask ECB, with the error and reason codes from the DYNALLOC request block, and a message is issued to give these values to the terminal.

DFH99DY then returns to the main program, which calls DFH99RP to process returned values. DFH99RP scans the returned value chain, and for each element issues a message containing the keyword and the value found in the text unit. If a returned value corresponds to a keyword value, DFH99KR is called to look up the value in the description, and issue the message.

Processing of the command is now complete, and the main program is reinitialized for the next one, and loops back to the point where it calls DFH99GI.

Messages are issued at many places, using macros. The macro expansion ends with a call to DFH99MP, which ensures that a new line is started for each new message, and calls DFH99ML, the message editor. Input to the message editor is a list of tokens, and each one is picked up in turn and converted to displayable text. For each piece of text, DFH99TX is called, which inserts the text into the output buffer, starting a new line if necessary. This ensures that a word is never split over two lines.

When the command has been processed, the main program calls DFH99MP with no parameters, which causes it to send the output buffer to the terminal, and initialize it to empty.

Control blocks

The sample program does not have any control blocks.

Modules

Module	Function
DFH99BC	Convert to binary target
DFH99CC	Character and number string conversion
DFH99DY	Issue SVC 99 and analyze result
DFH99FP	Process function keyword
DFH99GI	Format display and get input
DFH99KC	Keyword value conversion
DFH99KH	List keywords for help
DFH99KO	Process operator keywords
DFH99KR	Convert returned value to keyword
DFH99LK	Search key set for given token
DFH99ML	Build message text from token list
DFH99MM	Main control program (entry point DFH99M)
DFH99MP	Message filing routine
DFH99MT	Match abbreviation with keyword
DFH99RP	Process returned values
DFH99T	Table of keywords
DFH99TK	Tokenize input command
DFH99TX	Text display routine
DFH99VH	List description for help

Exits

No global user exit points are provided for this function.

Trace

This sample program makes no entries in the trace, over and above the normal entries one would see for a CICS user transaction.

External interfaces

SVC 99—MVS DYNALLOC SVC.

Chapter 18. ECI over TCP/IP

The IP ECI (IE) domain processes external call interface (ECI) requests that arrive from a CICS client that is connected to CICS by a TCP/IP network. It attaches a mirror task to issue the appropriate program link request, and returns the results to the client.

Design Overview

The CICS code that processes external call interface (ECI) requests that arrive from a TCP/IP network via the Sockets Domain (SO) is mostly contained within the IP ECI (IE) domain. Some code that is logically part of the function runs in AP domain. This is because SO domain works by attaching a listener task (CIEP for IPECI) to handle incoming data, and IE domain attaches a mirror task (CPMI) to issue the program link request and return any resulting output.

There are five logically separate pieces of code for this function:

- IE domain initialisation and termination code in DFHIEDM. See Chapter 84, “IP ECI (IE) domain,” on page 811 for more information.
- The AP domain part of the listener task, in program DFHIEP.
- The IE domain part of the listener task, in the PROCESS_ECI_FLOW function of program DFHIEIE. See Chapter 84, “IP ECI (IE) domain,” on page 811 for more information.
- The AP domain part of the mirror task, in programs DFHMIRS and DFHIEXM.
- The IE domain part of the mirror task, in the SEND, RECEIVE and SEND_ERROR functions of program DFHIEIE. See Chapter 84, “IP ECI (IE) domain,” on page 811 for more information.

Listener task, CIEP

The CIEP task is attached by SO domain when it receives data on the port specified in the IPECI TCPIPSERVICE. The CIEP transaction handles control flows directly, or attaches a mirror task to issue the ECI program link request.

The valid flows that may be received by CIEP are:

- Attach FMH for CCIN INSTALL

  The initial flow from a client is an attach for the CCIN transaction to install the client. No attach is done as IE domain handles the install processing internally.
- Attach FMH for CCIN UNINSTALL

  A client can terminate its connection with CICS by sending a CCIN UNINSTALL transaction request. No attach is done as IE domain handles the install processing internally.
- Attach FMH for some other transid, assumed to be a mirror
- FMH7 indicating the client wishes to abend a conversation.
- Connection level PING request/reply
- Conversation level PING request/reply
- Connection status 01, last transmission from client (equivalent to UNINSTALL)
- User data in extended conversation (Link request or SYNCPOINT RU)

All other flows are rejected by CIEP; conversation errors with an FMH7, control errors by closing the socket.

The different flows are distinguished by testing various fields in the flow headers, including the SNA format RH.

Request header settings

Response headers are never sent. All flows have request headers. Errors are returned by sending FMH7 with CEB.

All flows are OIC,RQE1.

The link requests to a long running mirror are packaged as FMH43s but, because they are within a GDS, should not cause the RH FMH bit to be set on.

Direction	Type of flow	Request header flags					
in	CCIN INSTALL FMH5	BB		OIC	CD	RQE1	FMH
out	CCIN INSTALL reply		CEB	OIC		RQE1	
in	CCIN UNINSTALL request	BB	CEB	OIC		RQE1	FMH
in	Mirror FMH5 + link request	BB		OIC	CD	RQE1	FMH
out	Non long-running mirror link reply		CEB	OIC		RQE1	
out	Long-running mirror link reply			OIC	CD	RQE1	
in	Long-running mirror link request			OIC	CD	RQE1	
in	Long-running mirror sync flow			OIC	CD	RQE1	
out	Long-running mirror sync reply		CEB	OIC		RQE1	
out	Conversation failure (FMH7)		CEB	OIC		RQE1	FMH
in	FMH7		CEB	OIC		RQE1	FMH

Mirror task, CPMI

A mirror task is attached by the listener task to handle a particular client conversation. The transaction attach callback module for IE mirrors is DFHIEXM. It sets the IECCB (IP ECI Conversation Control Block) to be the mirror task's facility token and establishes security context for the mirror task, using userid and password sent from the client where required.

The mirror task main program, DFHMIRS, issues the IEIE RECEIVE for the available data, and then performs the same functions as it does for ECI requests received in other environments. It then issues the IEIE SEND to return the output from the linked program to the client. For a conversation marked by the client as 'extended', the mirror then issues another IEIE RECEIVE which causes it to be suspended, waiting for more data. For a non-extended conversation, the mirror terminates after the SEND.

PING

CICS TS 2.2 supports full connection and conversation level PING as architected for the CICS family. This consists of defined flows to allow CICS to determine whether specified connections, or particular conversations on a connection, should still be considered active. CICS TS sends a PING request if the RTIMOUT interval is exceeded when waiting for data from a client:

- Send conversation level PING if the client install indicated this was supported.
- Send connection level PING otherwise.
- If it is a conversation PING that has timed out abend the task after sending a connection level PING to see if the client is still active.
- If a connection level PING times out, uninstall the client.

Notes

1. The socket is full duplex, so SENDs and RECEIVEs can be issued in any order, and asynchronously by different CICS tasks. This is necessary for multiple conversations on the same socket, and means that the CIEP task can issue a SOCK RECEIVE as soon as it has attached the mirror. The SOCK SEND will be done under the mirror task.
2. Sending tasks ENQ on the socket to prevent the data from multiple conversations being interleaved. The ENQ is issued by SO domain.
3. The SO socket token is the second part of the user token but is never required in the CIEP task. The sends and receives issued from CIEP use the socket implicit in the task's state.
4. If the connection is lost or closed by TCP/IP and there are long running mirrors waiting on receives, SO domain is notified, attaches CIEP and returns a bad response on the SO receive issued by CIEP.

Modules

DFHIEP

The initial program for the IP ECI listener transaction, CIEP.

DFHIEXM

The IPECI mirror transaction attach callback module.

Sets the IECCB to be the mirror task's facility token.

Establishes security context for the mirror task, using userid and password sent from client where required.

Chapter 19. EXEC interface

The EXEC interface provides the support for application programs containing EXEC CICS commands.

Design overview

The relevant parts of the EXEC interface are:

- The main EXEC interface module, DFHEIP, which is called when an EXEC CICS command is executed in a user application program.

  A parameter list is passed, in which the first argument (referred to as arg-zero) contains a group code and a function code as the first 2 bytes.

  – The group code in general indicates the CICS component associated with the command being executed. In subsequent processing it is this code alone which determines which EXEC processor module (see below) is called from DFHEIP.

  – The function code identifies the actual command being executed.

  Note: DFHEIP is link-edited with other modules to form the application interface program (DFHAIP) load module. DFHEIPA (next to be described) is one of these modules.

- The DFHEIPA module, which handles the allocation and freeing of dynamic storage (mapped by DFHEISTG) for assembler-language application programs in response to DFHEIENT and DFHEIRET calls respectively.
- A set of EXEC processor modules, each of which is called from DFHEIP, and which performs the first level of analysis of the command being executed. The processor then calls the appropriate CICS domain to complete the execution of the command.
- A set of EXEC stubs, one for each of the application languages: COBOL, PL/I, C, and assembler language. The appropriate stub must be link-edited at the front of each CICS application program, and provides the mechanism for getting to the correct entry points in DFHEIP.
- The DFHAPLI module, which is called at the initialization and termination of each application program.

Control blocks

The control blocks associated with the EXEC interface are as follows:

EXEC interface block (EIB) (DSECT name: DFHEIBLK).

Each task in a command-level environment has a control block called the EXEC interface block (EIB) associated with it. The EIB is used for direct communication between command-level programs and CICS.

The EIB contains information that is useful during the execution of an application program, such as the transaction identifier, the time and date (initially when the task is started, and subsequently, if updated by the application program), and the cursor position on a display device. The EIB also contains information that is helpful when a dump is being used to debug a program. DFHEIBLK defines the layout of an EIB, and is included automatically in the application program, giving access to all of the fields in the EIB by name.

A further EIB, known as the "system" EIB, exists for each task. The system EIB has the same format as the "user" (or "application") EIB. It is intended for use mainly by CICS system code. In general, application programs have addressability to the user EIB only, which is a copy taken of the system EIB at appropriate times. However, any service programs translated with the SYSEIB option have addressability to the system EIB also, so that they can issue EXEC CICS commands without causing the user EIB to be updated. (See the *CICS Application Programming Guide* for further information about the SYSEIB translator option.)

Figure 43 shows the format of an EIB.

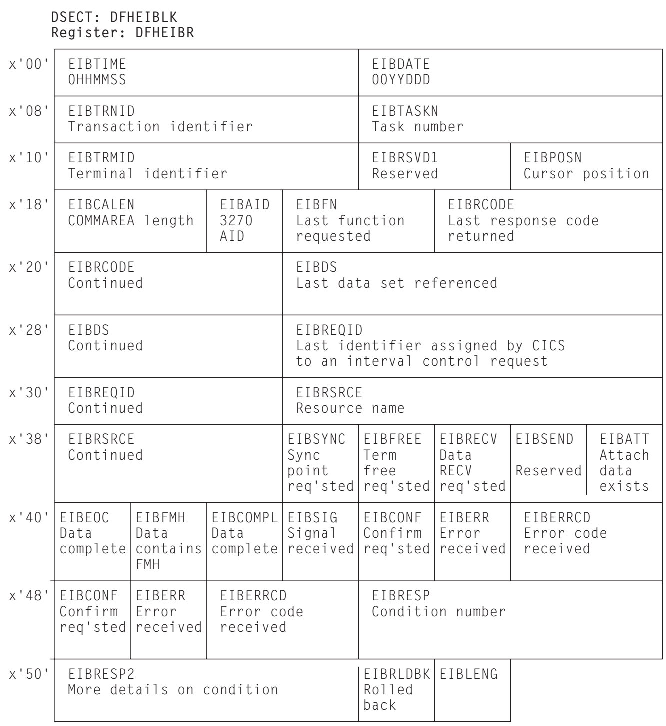
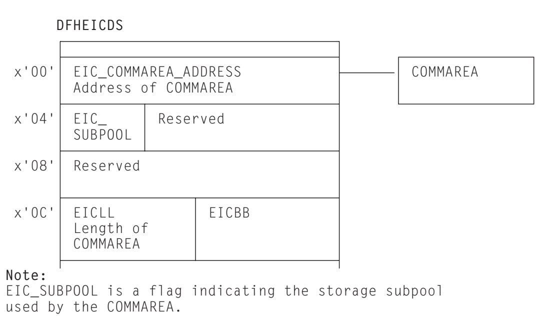


Figure 43. EXEC interface block (EIB)

EXEC interface communication area (DSECT name: DFHEICDS).

The EXEC interface communication area describes the storage that is used to pass the COMMAREA from one command-level transaction to another using an EXEC CICS RETURN command with the TRANSID, COMMAREA, and LENGTH options.

Figure 44 shows the format of the EXEC interface communication area.



Note: EIC_SUBPOOL is a flag indicating the storage subpool used by the COMMAREA.

Figure 44. EXEC interface communication area (EIC)

EXEC interface storage (EIS) (DSECT name: DFHEISDS).

The EXEC interface storage is used by DFHEIP as the interface between the application program and CICS control blocks. It contains a system area used by DFHEIP only. EIS is storage acquired by the DFHAPXM module (part of the transaction manager), along with other task-lifetime storage such as the TCA and both system and user EIBs. There is one EIS per transaction (not per program), and it is addressed by TCAEISA in the TCA. (See Figure 45.)

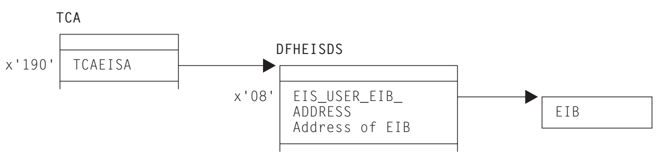


Figure 45. EXEC interface storage (EIS)

See the *CICS Data Areas* manual for a detailed description of these control blocks.

Modules

The EXEC interface comprises the following modules:
- The main interface module (DFHEIP)
- Prologue and epilogue code for assembler-language programs (DFHEIPA)
- 55 EXEC interface processors
- 4 EXEC stubs.

Of the EXEC interface processors, 16 are coded in Assembler language; the other modules are coded in other languages. All are CICS nucleus modules.

These processor modules (together with DFHEIP) support the EXEC CICS commands listed in Table 6.

DFHEIP also supports EXEC DLI commands, by passing them through the external resource manager interface program, DFHERM, on their way to DFHEDP for conversion to standard CALL parameter lists acceptable to DL/I.

The following tables list all the EXEC CICS commands, showing the class of each command (basic or special), its group and function codes, also the name and language of the associated EXEC interface processor. Table 6 is ordered by command name. Chapter 82, “Enterprise Java domain (EJ),” on page 753 is ordered by group/function code.

The group and function codes used by the Front End Programming Interface (FEPI) feature are not listed in these tables. However, the EXEC CICS FEPI commands use group codes of 82 (API-type commands) and 84 (SPI-type commands). For details about the EXEC CICS FEPI commands, see the *CICS Front End Programming Interface User's Guide*.

Note: An asterisk (*) after a command name in the tables shows that the command is intended for CICS internal use only.

Table 6. EXEC CICS commands ordered by command name

Command	Class	Gp/fn code	Module DFH...	Lang
ABEND	B	0E 0C	EPC	A
ACQUIRE TERMINAL	S	86 02	EIACQ	O
ADDRESS	B	02 02	EEI	A
ADDRESS SET	B	02 10	EEI	A
ALLOCATE	B	04 20	ETC	A
ASKTIME	B	10 02	EIIC	O

Table 6. EXEC CICS commands ordered by command name (continued)

Command	Class	Gp/fn code	Module DFH...	Lang
ASKTIME ABSTIME	B	4A 02	EIDTI	O
ASSIGN	B	02 08	EEI	A
BIF DEEDIT	B	20 02	EBF	A
BUILD ATTACH	B	04 26	ETC	A
CANCEL	B	10 0C	EIIC	O
CHANGE TASK	B	5E 06	EIQSK	O
COLLECT STATISTICS	S	70 08	EIQMS	O
CONNECT PROCESS	B	04 32	ETC	A
CONVERSE	B	04 06	ETC	A
CREATE CONNECTION	S	30 0E	EICRE	O
CREATE FILE	S	30 14	EICRE	O
CREATE JOURNALMODEL	S	30 1E	EICRE	O
CREATE LSRPOOL	S	30 16	EICRE	O
CREATE MAPSET	S	30 04	EICRE	O
CREATE PARTITIONSET	S	30 06	EICRE	O
CREATE PARTNER	S	30 18	EICRE	O
CREATE PROFILE	S	30 0A	EICRE	O
CREATE PROGRAM	S	30 02	EICRE	O
CREATE SESSIONS	S	30 12	EICRE	O
CREATE TDQUEUE	S	30 1C	EICRE	O
CREATE TERMINAL	S	30 10	EICRE	O
CREATE TRANCLASS	S	30 1A	EICRE	O
CREATE TRANSACTION	S	30 08	EICRE	O
CREATE TYPETERM	S	30 0C	EICRE	O
DELAY	B	10 04	EIIC	O
DELETE	B	06 08	EIFC	O
DELETEQ TD	B	08 06	ETD	A
DELETEQ TS	B	0A 06	ETS	A
DEQ	B	12 06	EKC	A
DISCARD AUTINSTMODEL	S	42 10	EIQTM	O
DISCARD FILE	S	4C 10	EIQDS	O
DISCARD JOURNALMODEL	S	92 10	EIQSL	O
DISCARD JOURNALNAME	S	60 10	EIQSJ	O
DISCARD PARTNER	S	44 10	EIQPN	O
DISCARD PROFILE	S	46 10	EIQPF	O
DISCARD PROGRAM	S	4E 10	EIQSP	O
DISCARD TRANSACTION	S	50 10	EIQSX	O
DISABLE	B	22 04	UEM	A
DUMP	B	1C 02	EDC	A
DUMP SYSTEM	B	7E 04	EDCP	O
DUMP TRANSACTION	B	7E 02	EDCP	O
ENABLE	B	22 02	UEM	A
ENDBR	B	06 12	EIFC	O
ENQ	B	12 04	EKC	A
ENTER TRACEID	B	1A 04	ETR	A
ENTER TRACENUM	B	48 02	ETRX	O
EXTRACT ATTACH	B	04 28	ETC	A
EXTRACT ATTRIBUTES	B	04 3E	ETC	A
EXTRACT EXIT	B	22 06	UEM	A
EXTRACT LOGONMSG	B	04 3C	ETC	A
EXTRACT PROCESS	B	04 2E	ETC	A
EXTRACT TCT	B	04 2A	ETC	A

Table 6. EXEC CICS commands ordered by command name (continued)

Command	Class	Gp/fn code	Module DFH...	Lang
FORMATTIME	B	4A 04	EIDTI	O
FREE	B	04 22	ETC	A
FREEMAIN	B	0C 04	ESC	A
GDS ALLOCATE	B	24 02	EGL	A
GDS ASSIGN	B	24 04	EGL	A
GDS CONNECT PROCESS	B	24 0C	EGL	A
GDS EXTRACT ATTRIBUTES	B	24 1C	EGL	A
GDS EXTRACT PROCESS	B	24 06	EGL	A
GDS FREE	B	24 08	EGL	A
GDS ISSUE ABEND	B	24 0A	EGL	A
GDS ISSUE CONFIRMATION	B	24 0E	EGL	A
GDS ISSUE ERROR	B	24 10	EGL	A
GDS ISSUE PREPARE	B	24 1A	EGL	A
GDS ISSUE SIGNAL	B	24 12	EGL	A
GDS RECEIVE	B	24 14	EGL	A
GDS SEND	B	24 16	EGL	A
GDS WAIT	B	24 18	EGL	A
GETMAIN	B	0C 02	ESC	A
HANDLE ABEND	B	0E 0E	EPC	A
HANDLE AID	B	02 06	EEI	A
HANDLE CONDITION	B	02 04	EEI	A
IGNORE CONDITION	B	02 0A	EEI	A
INQUIRE AUTINSTMODEL	S	42 02	EIQTM	O
INQUIRE AUTOINSTALL	S	68 12	EIQVT	O
INQUIRE CONNECTION	S	58 02	EIQSC	O
INQUIRE DCE	S	8E 02	EIQDE	O
INQUIRE DSNAME	S	7A 02	EIQDN	O
INQUIRE DUMPDS	S	66 02	EIQDU	O
INQUIRE EXITPROGRAM	S	88 02	EIQUE	O
INQUIRE FILE	S	4C 02	EIQDS	O
INQUIRE IRC	S	6E 02	EIQIR	O
INQUIRE JOURNALMODEL	S	92 02	EIQSL	O
INQUIRE JOURNALNAME	S	60 12	EIQSJ	O
INQUIRE JOURNALNUM	S	60 02	EIQSJ	O
INQUIRE MODENAME	S	5A 02	EIQSM	O
INQUIRE MONITOR	S	70 12	EIQMS	O
INQUIRE NETNAME	S	52 06	EIQST	O
INQUIRE PARTNER	S	44 02	EIQPN	O
INQUIRE PROFILE	S	46 02	EIQPF	O
INQUIRE PROGRAM	S	4E 02	EIQSP	O
INQUIRE REQID	S	8A 02	EIQRQ	O
INQUIRE STATISTICS	S	70 02	EIQMS	O
INQUIRE STREAMNAME	S	92 12	EIQSL	O
INQUIRE SYSDUMPCODE	S	66 22	EIQDU	O
INQUIRE SYSTEM	S	54 02	EIQSA	O
INQUIRE TASK	S	5E 02	EIQSK	O
INQUIRE TCLASS	S	5E 12	EIQSK	O
INQUIRE TDQUEUE	S	5C 02	EIQSQ	O
INQUIRE TERMINAL	S	52 02	EIQST	O
INQUIRE TRACEDEST	S	78 02	EIQTR	O
INQUIRE TRACEFLAG	S	78 12	EIQTR	O
INQUIRE TRACETYPE	S	78 22	EIQTR	O

Table 6. EXEC CICS commands ordered by command name (continued)

Command	Class	Gp/fn code	Module DFH...	Lang
INQUIRE TRANDUMPCODE	S	66 12	EIQDU	O
INQUIRE TRANSACTION	S	50 02	EIQSX	O
INQUIRE TSQUEUE	S	0A 08	EIQTS	O
INQUIRE VTAM	S	68 02	EIQVT	O
ISSUE ABEND	B	04 30	ETC	A
ISSUE ABORT	B	1E 08	EDI	A
ISSUE ADD	B	1E 02	EDI	A
ISSUE CONFIRMATION	B	04 34	ETC	A
ISSUE COPY	B	04 0A	ETC	A
ISSUE DISCONNECT	B	04 14	ETC	A
ISSUE END	B	1E 0C	EDI	A
ISSUE ENDFILE	B	04 1A	ETC	A
ISSUE ENDOUTPUT	B	04 16	ETC	A
ISSUE EODS	B	04 08	ETC	A
ISSUE ERASE	B	1E 04	EDI	A
ISSUE ERASEAUP	B	04 18	ETC	A
ISSUE ERROR	B	04 36	ETC	A
ISSUE LOAD	B	04 0E	ETC	A
ISSUE NOTE	B	1E 10	EDI	A
ISSUE PASS	B	04 3A	ETC	A
ISSUE PREPARE	B	04 38	ETC	A
ISSUE PRINT	B	04 1C	ETC	A
ISSUE QUERY	B	1E 0A	EDI	A
ISSUE RECEIVE	B	1E 0E	EDI	A
ISSUE REPLACE	B	1E 06	EDI	A
ISSUE RESET	B	04 12	ETC	A
ISSUE SEND	B	1E 14	EDI	A
ISSUE SIGNAL	B	04 1E	ETC	A
ISSUE WAIT	B	1E 12	EDI	A
LINK	B	0E 02	EPC	A
LOAD	B	0E 06	EPC	A
MONITOR	B	48 04	ETRX	O
PERFORM RESETTIME	S	72 02	EIPRT	O
PERFORM SECURITY	S	64 02	EIPSE	O
PERFORM SHUTDOWN	S	76 02	EIPSH	O
PERFORM STATISTICS	S	70 06	EIQMS	O
POINT	B	04 24	ETC	A
POP	B	02 0E	EEI	A
POST	B	10 06	EIIC	O
PURGE MESSAGE	B	18 0A	EMS	A
PUSH	B	02 0C	EEI	A
QUERY SECURITY	B	6A 02	ESE	O
READ	B	06 02	EIFC	O
READNEXT	B	06 0E	EIFC	O
READPREV	B	06 10	EIFC	O
READQ TD	B	08 04	ETD	A
READQ TS	B	0A 04	ETS	A
RECEIVE	B	04 02	ETC	A
RECEIVE MAP	B	18 02	EMS	A
RECEIVE PARTN	B	18 0E	EMS	A
RELEASE	B	0E 0A	EPC	A
RESETBR	B	06 14	EIFC	O

Table 6. EXEC CICS commands ordered by command name (continued)

Command	Class	Gp/fn code	Module DFH...	Lang
RESYNC	B	16 04	ESP	A
RETRIEVE	B	10 0A	EIIC	O
RETURN	B	0E 08	EPC	A
REWRITE	B	06 06	EIFC	O
ROUTE	B	18 0C	EMS	A
SEND	B	04 04	ETC	A
SEND CONTROL	B	18 12	EMS	A
SEND MAP	B	18 04	EMS	A
SEND PAGE	B	18 08	EMS	A
SEND PARTNSET	B	18 10	EMS	A
SEND TEXT	B	18 06	EMS	A
SET AUTOINSTALL	S	68 14	EIQVT	O
SET CONNECTION	S	58 04	EIQSC	O
SET DCE	S	8E 04	EIQDE	O
SET DSNAME	S	7A 04	EIQDN	O
SET DUMPDS	S	66 04	EIQDU	O
SET FILE	S	4C 04	EIQDS	O
SET IRC	S	6E 04	EIQIR	O
SET JOURNALNAME	S	60 14	EIQSJ	O
SET JOURNALNUM	S	60 04	EIQSJ	O
SET MODENAME	S	5A 04	EIQSM	O
SET MONITOR	S	70 14	EIQMS	O
SET NETNAME	S	52 08	EIQST	O
SET PROGRAM	S	4E 04	EIQSP	O
SET STATISTICS	S	70 04	EIQMS	O
SET SYSDUMPCODE	S	66 24	EIQDU	O
SET SYSTEM	S	54 04	EIQSA	O
SET TASK	S	5E 04	EIQSK	O
SET TCLASS	S	5E 14	EIQSK	O
SET TDQUEUE	S	5C 04	EIQSQ	O
SET TERMINAL	S	52 04	EIQST	O
SET TRACEDEST	S	78 04	EIQTR	O
SET TRACEFLAG	S	78 14	EIQTR	O
SET TRACETYPE	S	78 24	EIQTR	O
SET TRANDUMPCODE	S	66 14	EIQDU	O
SET TRANSACTION	S	50 04	EIQSX	O
SET VTAM	S	68 04	EIQVT	O
SIGNOFF	B	74 04	ESN	O
SIGNON	B	74 02	ESN	O
SPOOLCLOSE	B	56 10	EPS	O
SPOOLOPEN	B	56 02	EPS	O
SPOOLREAD	B	56 04	EPS	O
SPOOLWRITE	B	56 06	EPS	O
START	B	10 08	EIIC	O
STARTBR	B	06 0C	EIFC	O
SUSPEND	B	12 08	EKC	A
SYNCPOINT	B	16 02	ESP	A
TRACE	B	1A 02	ETR	A
UNLOCK	B	06 0A	EIFC	O
WAIT CONVID	B	04 2C	ETC	A
WAIT EVENT	B	12 02	EKC	A
WAIT EXTERNAL	B	5E 22	EIQSK	O

Table 6. EXEC CICS commands ordered by command name (continued)

Command	Class	Gp/fn code	Module DFH...	Lang
WAIT JOURNALNAME	B	14 08	EJC	A
WAIT JOURNALNUM	B	14 04	EJC	A
WAIT SIGNAL	B	04 10	ETC	A
WAIT TERMINAL	B	04 0C	ETC	A
WAITCICS	B	5E 32	EIQSK	O
WRITE FILE	B	06 04	EIFC	O
WRITE JOURNALNAME	B	14 06	EJC	A
WRITE JOURNALNUM	B	14 02	EJC	A
WRITE OPERATOR	B	6C 02	EOP	O
WRITEQ TD	B	08 02	ETD	A
WRITEQ TS	B	0A 02	ETS	A
XCTL	B	0E 04	EPC	A

Abbreviations:

```
Class of command:    B = basic       S = special
Language of module:  A = assembler   O = other
```

Table 7. EXEC CICS commands ordered by group/function code

Command	Class	Gp/fn code	Module DFH...	Lang
ADDRESS	B	02 02	EEI	A
HANDLE CONDITION	B	02 04	EEI	A
HANDLE AID	B	02 06	EEI	A
ASSIGN	B	02 08	EEI	A
IGNORE CONDITION	B	02 0A	EEI	A
PUSH	B	02 0C	EEI	A
POP	B	02 0E	EEI	A
ADDRESS SET	B	02 10	EEI	A
RECEIVE	B	04 02	ETC	A
SEND	B	04 04	ETC	A
CONVERSE	B	04 06	ETC	A
ISSUE EODS	B	04 08	ETC	A
ISSUE COPY	B	04 0A	ETC	A
WAIT TERMINAL	B	04 0C	ETC	A
ISSUE LOAD	B	04 0E	ETC	A
WAIT SIGNAL	B	04 10	ETC	A
ISSUE RESET	B	04 12	ETC	A
ISSUE DISCONNECT	B	04 14	ETC	A
ISSUE ENDOUTPUT	B	04 16	ETC	A
ISSUE ERASEAUP	B	04 18	ETC	A
ISSUE ENDFILE	B	04 1A	ETC	A
ISSUE PRINT	B	04 1C	ETC	A
ISSUE SIGNAL	B	04 1E	ETC	A
ALLOCATE	B	04 20	ETC	A
FREE	B	04 22	ETC	A
POINT	B	04 24	ETC	A
BUILD ATTACH	B	04 26	ETC	A
EXTRACT ATTACH	B	04 28	ETC	A
EXTRACT TCT	B	04 2A	ETC	A
WAIT CONVID	B	04 2C	ETC	A
EXTRACT PROCESS	B	04 2E	ETC	A
ISSUE ABEND	B	04 30	ETC	A
CONNECT PROCESS	B	04 32	ETC	A

Table 7. EXEC CICS commands ordered by group/function code (continued)

Command	Class	Gp/fn code	Module DFH...	Lang
ISSUE CONFIRMATION	B	04 34	ETC	A
ISSUE ERROR	B	04 36	ETC	A
ISSUE PREPARE	B	04 38	ETC	A
ISSUE PASS	B	04 3A	ETC	A
EXTRACT LOGONMSG	B	04 3C	ETC	A
EXTRACT ATTRIBUTES	B	04 3E	ETC	A
READ	B	06 02	EIFC	O
WRITE FILE	B	06 04	EIFC	O
REWRITE	B	06 06	EIFC	O
DELETE	B	06 08	EIFC	O
UNLOCK	B	06 0A	EIFC	O
STARTBR	B	06 0C	EIFC	O
READNEXT	B	06 0E	EIFC	O
READPREV	B	06 10	EIFC	O
ENDBR	B	06 12	EIFC	O
RESETBR	B	06 14	EIFC	O
WRITEQ TD	B	08 02	ETD	A
READQ TD	B	08 04	ETD	A
DELETEQ TD	B	08 06	ETD	A
WRITEQ TS	B	0A 02	ETS	A
READQ TS	B	0A 04	ETS	A
DELETEQ TS	B	0A 06	ETS	A
INQUIRE TSQUEUE	S	0A 08	EIQTS	O
GETMAIN	B	0C 02	ESC	A
FREEMAIN	B	0C 04	ESC	A
LINK	B	0E 02	EPC	A
XCTL	B	0E 04	EPC	A
LOAD	B	0E 06	EPC	A
RETURN	B	0E 08	EPC	A
RELEASE	B	0E 0A	EPC	A
ABEND	B	0E 0C	EPC	A
HANDLE ABEND	B	0E 0E	EPC	A
ASKTIME	B	10 02	EIIC	O
DELAY	B	10 04	EIIC	O
POST	B	10 06	EIIC	O
START	B	10 08	EIIC	O
RETRIEVE	B	10 0A	EIIC	O
CANCEL	B	10 0C	EIIC	O
WAIT EVENT	B	12 02	EKC	A
ENQ	B	12 04	EKC	A
DEQ	B	12 06	EKC	A
SUSPEND	B	12 08	EKC	A
WRITE JOURNALNUM	B	14 02	EJC	A
WAIT JOURNALNUM	B	14 04	EJC	A
SYNCPOINT	B	16 02	ESP	A
RESYNC	B	16 04	ESP	A
RECEIVE MAP	B	18 02	EMS	A
SEND MAP	B	18 04	EMS	A
SEND TEXT	B	18 06	EMS	A
SEND PAGE	B	18 08	EMS	A
PURGE MESSAGE	B	18 0A	EMS	A
ROUTE	B	18 0C	EMS	A

Table 7. EXEC CICS commands ordered by group/function code (continued)

Command	Class	Gp/fn code	Module DFH...	Lang
RECEIVE PARTN	B	18 0E	EMS	A
SEND PARTNSET	B	18 10	EMS	A
SEND CONTROL	B	18 12	EMS	A
TRACE	B	1A 02	ETR	A
ENTER TRACEID	B	1A 04	ETR	A
DUMP	B	1C 02	EDC	A
ISSUE ADD	B	1E 02	EDI	A
ISSUE ERASE	B	1E 04	EDI	A
ISSUE REPLACE	B	1E 06	EDI	A
ISSUE ABORT	B	1E 08	EDI	A
ISSUE QUERY	B	1E 0A	EDI	A
ISSUE END	B	1E 0C	EDI	A
ISSUE RECEIVE	B	1E 0E	EDI	A
ISSUE NOTE	B	1E 10	EDI	A
ISSUE WAIT	B	1E 12	EDI	A
ISSUE SEND	B	1E 14	EDI	A
BIF DEEDIT	B	20 02	EBF	A
ENABLE	B	22 02	UEM	A
DISABLE	B	22 04	UEM	A
EXTRACT EXIT	B	22 06	UEM	A
GDS ALLOCATE	B	24 02	EGL	A
GDS ASSIGN	B	24 04	EGL	A
GDS EXTRACT PROCESS	B	24 06	EGL	A
GDS FREE	B	24 08	EGL	A
GDS ISSUE ABEND	B	24 0A	EGL	A
GDS CONNECT PROCESS	B	24 0C	EGL	A
GDS ISSUE CONFIRMATION	B	24 0E	EGL	A
GDS ISSUE ERROR	B	24 10	EGL	A
GDS ISSUE SIGNAL	B	24 12	EGL	A
GDS RECEIVE	B	24 14	EGL	A
GDS SEND	B	24 16	EGL	A
GDS WAIT	B	24 18	EGL	A
GDS ISSUE PREPARE	B	24 1A	EGL	A
GDS EXTRACT ATTRIBUTES	B	24 1C	EGL	A
CREATE PROGRAM	S	30 02	EICRE	O
CREATE MAPSET	S	30 04	EICRE	O
CREATE PARTITIONSET	S	30 06	EICRE	O
CREATE TRANSACTION	S	30 08	EICRE	O
CREATE PROFILE	S	30 0A	EICRE	O
CREATE TYPETERM	S	30 0C	EICRE	O
CREATE CONNECTION	S	30 0E	EICRE	O
CREATE TERMINAL	S	30 10	EICRE	O
CREATE SESSIONS	S	30 12	EICRE	O
CREATE FILE	S	30 14	EICRE	O
CREATE LSRPOOL	S	30 16	EICRE	O
CREATE PARTNER	S	30 18	EICRE	O
CREATE TRANCLASS	S	30 1A	EICRE	O
CREATE TDQUEUE	S	30 1C	EICRE	O
CREATE JOURNALMODEL	S	30 1E	EICRE	O
INQUIRE AUTINSTMODEL	S	42 02	EIQTM	O
DISCARD AUTINSTMODEL	S	42 10	EIQTM	O
INQUIRE PARTNER	S	44 02	EIQPN	O

Table 7. EXEC CICS commands ordered by group/function code (continued)

Command	Class	Gp/fn code	Module DFH...	Lang
DISCARD PARTNER	S	44 10	EIQPN	O
INQUIRE PROFILE	S	46 02	EIQPF	O
DISCARD PROFILE	S	46 10	EIQPF	O
ENTER TRACENUM	B	48 02	ETRX	O
MONITOR	B	48 04	ETRX	O
ASKTIME ABSTIME	B	4A 02	EIDTI	O
FORMATTIME	B	4A 04	EIDTI	O
INQUIRE FILE	S	4C 02	EIQDS	O
SET FILE	S	4C 04	EIQDS	O
DISCARD FILE	S	4C 10	EIQDS	O
INQUIRE PROGRAM	S	4E 02	EIQSP	O
SET PROGRAM	S	4E 04	EIQSP	O
DISCARD PROGRAM	S	4E 10	EIQSP	O
INQUIRE TRANSACTION	S	50 02	EIQSX	O
SET TRANSACTION	S	50 04	EIQSX	O
DISCARD TRANSACTION	S	50 10	EIQSX	O
INQUIRE TERMINAL	S	52 02	EIQST	O
SET TERMINAL	S	52 04	EIQST	O
INQUIRE NETNAME	S	52 06	EIQST	O
SET NETNAME	S	52 08	EIQST	O
INQUIRE SYSTEM	S	54 02	EIQSA	O
SET SYSTEM	S	54 04	EIQSA	O
SPOOLOPEN	B	56 02	EPS	O
SPOOLREAD	B	56 04	EPS	O
SPOOLWRITE	B	56 06	EPS	O
SPOOLCLOSE	B	56 10	EPS	O
INQUIRE CONNECTION	S	58 02	EIQSC	O
SET CONNECTION	S	58 04	EIQSC	O
INQUIRE MODENAME	S	5A 02	EIQSM	O
SET MODENAME	S	5A 04	EIQSM	O
INQUIRE TDQUEUE	S	5C 02	EIQSQ	O
SET TDQUEUE	S	5C 04	EIQSQ	O
INQUIRE TASK	S	5E 02	EIQSK	O
SET TASK	S	5E 04	EIQSK	O
CHANGE TASK	B	5E 06	EIQSK	O
INQUIRE TCLASS	S	5E 12	EIQSK	O
SET TCLASS	S	5E 14	EIQSK	O
WAIT EXTERNAL	B	5E 22	EIQSK	O
WAITCICS	B	5E 32	EIQSK	O
INQUIRE JOURNALNUM	S	60 02	EIQSJ	O
SET JOURNALNUM	S	60 04	EIQSJ	O
INQUIRE JOURNALNAME	S	60 12	EIQSJ	O
SET JOURNALNAME	S	60 14	EIQSJ	O
PERFORM SECURITY	S	64 02	EIPSE	O
INQUIRE DUMPDS	S	66 02	EIQDU	O
SET DUMPDS	S	66 04	EIQDU	O
INQUIRE TRANDUMPCODE	S	66 12	EIQDU	O
SET TRANDUMPCODE	S	66 14	EIQDU	O
INQUIRE SYSDUMPCODE	S	66 22	EIQDU	O
SET SYSDUMPCODE	S	66 24	EIQDU	O
INQUIRE VTAM	S	68 02	EIQVT	O
SET VTAM	S	68 04	EIQVT	O

Table 7. EXEC CICS commands ordered by group/function code (continued)

Command	Class	Gp/fn code	Module DFH...	Lang
INQUIRE AUTOINSTALL	S	68 12	EIQVT	O
SET AUTOINSTALL	S	68 14	EIQVT	O
QUERY SECURITY	B	6A 02	ESE	O
WRITE OPERATOR	B	6C 02	EOP	O
CICSMESSAGE *	S	6C 12	EOP	O
INQUIRE IRC	S	6E 02	EIQIR	O
SET IRC	S	6E 04	EIQIR	O
INQUIRE STATISTICS	S	70 02	EIQMS	O
SET STATISTICS	S	70 04	EIQMS	O
PERFORM STATISTICS	S	70 06	EIQMS	O
COLLECT STATISTICS	S	70 08	EIQMS	O
INQUIRE MONITOR	S	70 12	EIQMS	O
SET MONITOR	S	70 14	EIQMS	O
PERFORM RESETTIME	S	72 02	EIPRT	O
SIGNON	B	74 02	ESN	O
SIGNOFF	B	74 04	ESN	O
PERFORM SHUTDOWN	S	76 02	EIPSH	O
INQUIRE TRACEDEST	S	78 02	EIQTR	O
SET TRACEDEST	S	78 04	EIQTR	O
INQUIRE TRACEFLAG	S	78 12	EIQTR	O
SET TRACEFLAG	S	78 14	EIQTR	O
INQUIRE TRACETYPE	S	78 22	EIQTR	O
SET TRACETYPE	S	78 24	EIQTR	O
INQUIRE DSNAME	S	7A 02	EIQDN	O
SET DSNAME	S	7A 04	EIQDN	O
DUMP TRANSACTION	B	7E 02	EDCP	O
DUMP SYSTEM	B	7E 04	EDCP	O
INQUIRE JOURNALMODEL	S	92 02	EIQSL	O
INQUIRE STREAMNAME	S	92 12	EIQSL	O

Abbreviations:

```
Class of command:    B = basic       S = special
Language of module:  A = assembler   O = other
```

DFHEIP

The EXEC interface program, DFHEIP, has several entry points associated with initialization and termination. Note, however, that DFHEIPAN is in the DFHEIPA module.

Entry point
Function

DFHEIPNA
Formal main entry point

DFHEIPAN
Get or free dynamic storage for assembler-language prologue or epilogue

DFHEIPGM
Get dynamic storage for COBOL initialization

DFHEIPFM
Free dynamic storage for COBOL

DFHEIPTT
Take run-unit token routine for COBOL initialization.

DFHEIP has these entry points associated with executing a command issued from an application program:

Entry point
Function

DFHEIPRN
EXEC RMI calls

DFHEIPCN
EXEC CICS calls

DFHEIPDN
xxxTDLI calls.

It also has many return and entry points for common functions that are called from those processor modules residing in the nucleus:

Entry point
Function

EIPNORML
Normal return on completion of command

Error point
Function

EIPERROR
Condition occurred (code in EIBRCODE)

EIPCONDN
Condition occurred (code in EIBRESP)

EICCER99
Unsupported function, abend AEY9

EICCDF00
Subroutine to invoke EDF

Several length-checking routines (EICCLCnn):

Error point	Function
EICCLC30	Input check, V format only
EICCLC94	LENGERR flag check

Several program control routines (EICCPCnn):

Error point	Function
EICCPC00	Process terminating PL/I program
EICCPC40	HANDLE ABEND processing

Several storage control routines (EICCSCnn):

Error point	Function
EICCSC10	FREEMAIN
EICCSC20	GETMAIN shared storage
EICCSC30	GETMAIN terminal storage
EICCSC70	GETMAIN user storage init. X'00'
EICCFM10	FREEMAIN for COMMAREAs

Method of calling processor modules

All processor modules reside in the CICS nucleus, and the same calling method is used regardless of the language in which the processor is coded.

CICS initialization puts the address of each module in the CSA optional features list (CSAOPFL), in a table of addresses starting at CSAEXECS, and at an offset corresponding to its group code.

The calling method for the processor modules at execution time uses a table (at label EICC71T in DFHEIP), known as the **EXEC command processor module call table**. DFHEIP uses this table, and the table of addresses in CSAOPFL.

The EXEC command processor module call table is indexed by the 1-byte group code, which identifies the way that the processor is called:

Call type	Description
A	Has a vector of offsets at its entry point. This vector is indexed by the command function code to locate the actual entry point, to which DFHEIP does an unconditional branch. Return is to label EIPNORML, EIPCONDN, or EIPERROR.
B	Has a single entry point, for which DFHEIP issues a DFHAM TYPE=LINK call. The appropriate return address in DFHEIP is set in register 14, an unconditional branch is made to the DFHEIP, which tests the response in EIBRESP.
C	Has a single entry point, for which DFHEIP issues a DFHEIEIM call (through the kernel). Return is to the next instruction, where DFHEIP tests the response in EIBRESP.
D	Has a single entry point, for which DFHEIP uses a BALR R14,R15 instruction; this type is used only for GDS. The appropriate return address in DFHEIP is set in register 14, an unconditional branch is made to the DFHEIP, the response in the user's RETCODE field.

Exits

The following global user exit points are provided in DFHEIP:

XEIIN

XEIOUT

XEISPIN

XEISPOUT

For further information, see the *CICS Customization Guide*.

Trace

The following point ID is provided for DFHEIP:

- AP 00E1, for which the trace level is EI 1.

The following point IDs are provided for DFHEISR:

- AP E110 (entry), for which the trace level is EI 2.
- AP E111 (exit), for which the trace level is EI 2.

Trace entries are made before and after the execution of a command by its EXEC interface processor module.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 20. Execution diagnostic facility (EDF)

The execution diagnostic facility (EDF) allows users of the CICS command-level programming interface to step through the CICS commands of an application program. This program can be part of a local or remote transaction. At each step, the user can check the validity of each command and make temporary modifications to the program.

Design overview

EDF enables an application programmer to test a command-level application program online without making any modifications to the source program or the program preparation procedure. EDF intercepts execution of the application program at certain points and displays relevant information about the program at these points.

There are seven places in the EXEC interface program (DFHEIP) where the EDF can be called:

1. When program initialization has been done, just before control is passed to the application entry point
2. When program termination is being done, just after control has been received from the application
3. Before a normal EXEC command is passed to its processor module
4. When a normal EXEC command has returned to DFHEIP
5. Before an EXEC CICS GDS command is passed to its processor module
6. When an EXEC CICS GDS command has returned to DFHEIP
7. Before an EXEC CICS FEPI command is passed to its processor module
8. When an EXEC CICS FEPI command has returned to DFHEIP
9. At the end of a PL/I program.

Modules

CEBR transaction (DFHEDFBR)

The temporary-storage browse transaction (CEBR) allows the user to browse, copy, or delete items in a queue. CEBR invokes DFHEDFBR to execute the required action.

EDF display (DFHEDFD)

The EDF display program, DFHEDFD, provides the following functions:

- To display the user program status
- To allow the user to modify argument values and responses
- To allow the user to display and modify the EXEC interface block (EIB) and program working storage
- To allow the user to display any hexadecimal location in the partition user screen
- To allow the user to suppress EDF displays until specified conditions are met.

Method

1. Data describing user status is passed to DFHEDFD in the TWA.
2. Initialize exception and abend handling.
3. If TS queue for user terminal already exists, read control information; otherwise create control information about TS queue.
4. Check for security violation.
5. If necessary, remember user screen.
6. Build required display by calling DFHEDFS.
7. Send display to EDF screen.

8. Extract modified information by calling DFHEDFS.
9. Analyze request.
10. Set up build information for next display.
11. Go and build required display.
12. When no further displays are required:
    a. Save function display
    b. If necessary, restore user screen
    c. Update control information
    d. If transaction is defined as remote, purge TS queue and any shared storage associated with the EDF task
    e. Return to DFHEDFP.

EDF map set (DFHEDFM)

The EDF map set, DFHEDFM, consists of three maps:

DFHEDFM
To display status information at the various EDF interception points

DFHEDFN
To display the EDF stop conditions

DFHEDFP
To display a dump of storage.

All maps are (24,80). The first two lines of each map contain the transaction ID, program name, status, and so on. The format of these two lines must be identical for all maps. A menu is displayed with each map, and includes a message line and a reply field. The format of the menu must be identical for all maps. The cursor is positioned by symbolic cursor positioning.

EDF control program (DFHEDFP)

The EDF control program, DFHEDFP, provides the CEDF transaction for starting EDF, and is used in two different ways:
1. To control the debugging task
2. To set debug mode on or off.

Input

Input to the DFHEDFP program is provided as follows:

To control the debugging task
Information describing the user task status is written into the debug linkage area (DLA) of CEDF by DFHEDFX.

To set debug mode on or off
The user enters a CEDF request at the debug display terminal using the following syntax:

```
CEDF termid,ON|OFF
```

Alternatively, a PF key may be used to switch single-terminal debug mode on.

Note: To use EDF for a remote transaction, only single-terminal mode is available.

Output

Output from the DFHEDFP program is as follows:

To control the debugging task
DFHEDFD displays user program status.

To set debug mode on or off Switches the debug mode bit either in the user terminal TCTTE or, if an EXEC task is running, in the user task EIS. For two-terminal debugging, creates temporary-storage queue element to connect user terminal with display terminal.

Method

To control the program for debugging a task
If the task is attached by DFHEDFX and if only one terminal is being used for debugging, link to DFHEDFD to display program status. If two terminals are being used for debugging, start CEDF at the display terminal, restore that terminal to the user, resume the user task, then return to CICS.

To set debug mode on or off If invoked by using a PF key, set the debug mode on for single-terminal debugging in the user TCTTE. If invoked by a CEDF request, extract the user terminal ID (default is display terminal), and extract the debug mode (default is on). If the user terminal ID does not exist, output a diagnostic message. If the EXEC task is running and the task is in debug mode, output a diagnostic message; otherwise switch the debug bit in EIS, or switch the debug bit in TCTTE. Create a temporary-storage queue element naming the debug terminal.

EDF response table (DFHEDFR)

The EDF response table, DFHEDFR, is a table used by DFHEDFD to interpret the responses obtained by EXEC commands.

EDF task switch program (DFHEDFX)

The EDF task switch program, DFHEDFX, is used to attach the debugging task, provide it with all necessary information about the status of the user task, and suspend the user task until the debugging task allows it to resume.

Method

1. Extract information describing the user task status and copy it into the DLA for the attached task
2. Issue wait on user terminal
3. Attach CEDF
4. Suspend the user task
5. When the user task is resumed by EDF, check if EDF has not abended
6. If the user requests an abend, abend the user task; otherwise, return to caller.

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

Chapter 21. Extended recovery facility (XRF)

The extended recovery facility (XRF) enables you to achieve a high level of availability. You can run an alternate CICS system that monitors your active CICS system, and takes over automatically or by operator control if the active system fails. You can also plan and execute a takeover yourself when you want to do maintenance on an active system.

Problems in the active system can be detected and isolated as soon as they occur. The alternate system can recover and restart quickly, like an emergency restart, and the time for reconnection of terminals is reduced.

Design overview

A detailed overview of this function is given in the *CICS/ESA 3.3 CICS XRF Guide*.

Control blocks

A command list table (CLT) is used by an alternate system when it takes over the running of CICS from an active system. It holds the ID data for the JES system in use, data used to verify its authority to take over, and routing information. If there is more than one active system in two CECs, the CLT also holds VTAM MODIFY commands, and messages to the operator (WTO) to complete the takeover. It is loaded during takeover, and deleted when processed.

See the *CICS Data Areas* manual for a detailed description of this control block.

Modules

Figure 46 on page 156 shows the modules for XRF.

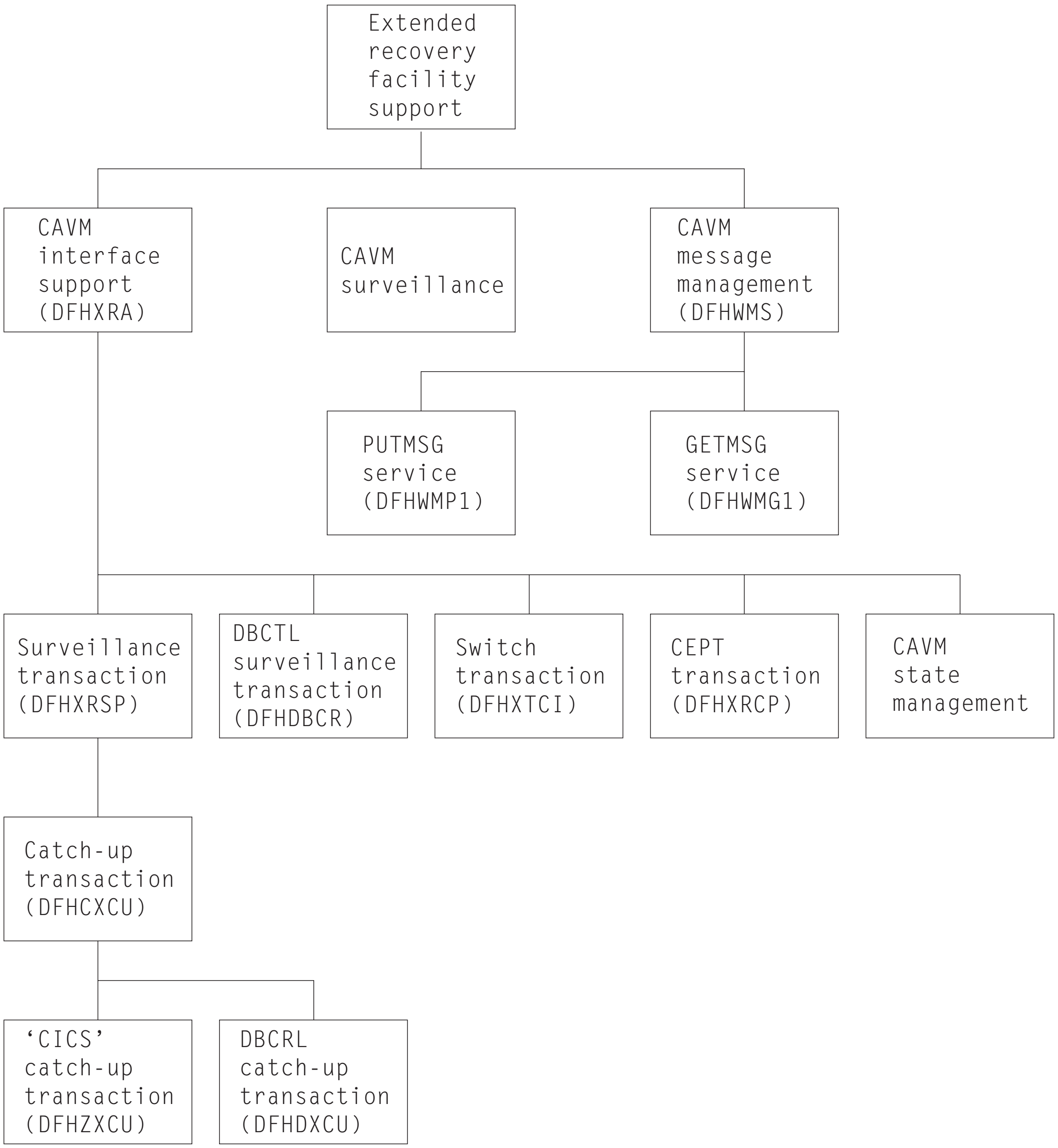


Figure 46. Extended recovery facility support

Exits

There is one global user exit point in DFHXRA: XXRSTAT. For further information about this, see the *CICS Customization Guide*.

Trace

The following point IDs are provided for the CAVM services:
- AP 00C4, AP 00C5, AP 00C6, and AP 00C7, for which the trace level is AP 1.

The following point IDs are provided for the XRF takeover signon/sign-off function:
- AP 0Axx, for which the trace levels are AP 1, AP 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 22. External CICS interface

The external CICS interface (EXCI) is an integral part of CICS Transaction Server for z/OS, Version 3 Release 1. The function is called an external CICS interface because it enables non-CICS application programs (*client programs*) running in MVS to call programs (*server programs*) running in a CICS Transaction Server for z/OS, Version 3 Release 1 region and to pass and receive data by means of a communications area.

Design overview

This section provides an overview of the design of the external CICS interface. For more information about the external CICS interface, see the *CICS External Interfaces Guide*.

The external CICS interface is an application programming interface that enables a non-CICS program (a *client program*) running in MVS to call a program (a *server program*) running in a CICS Transaction Server for z/OS, Version 3 Release 1 region and to pass and receive data by means of a communications area. The CICS application program is invoked as if linked-to by another CICS application program.

This programming interface allows a user to allocate and open sessions (or *pipes*[1]) to a CICS region, and to pass distributed program link (DPL) requests over them. The multiregion operation (MRO) facility of CICS interregion communication (IRC) facility supports these requests, and each pipe maps onto one MRO session.

Unless the CICS region is running in a sysplex under MVS/ESA 5.1 and therefore able to use cross-system MRO (XCF/MRO), the client program and the CICS server region (the region where the server program runs or is defined) must be in the same MVS image. Although the external CICS interface does not support the cross-memory access method, it can use the XCF access method provided by XCF/MRO in CICS Transaction Server for z/OS, Version 3 Release 1. See the CICS Intercommunication Guide for information about XCF/MRO.

A client program that uses the external CICS interface can operate multiple sessions for different users (either under the same or separate TCBs) all coexisting in the same MVS address space without knowledge of, or interference from, each other.

Where a client program attaches another client program, the attached program runs under its own TCB.

The programming interfaces

The external CICS interface provides two forms of programming interface: the EXCI CALL interface and the EXEC CICS interface.

The EXCI CALL interface

This interface consists of six commands that allow you to:

- Allocate and open sessions to a CICS system from non-CICS programs running under MVS
- Issue DPL requests on these sessions from the non-CICS programs
- Close and deallocate the sessions on completion of the DPL requests.

The six EXCI commands are:

1. Initialize_User
2. Allocate_Pipe
3. Open_Pipe
4. DPL call

1. **pipe**. A one-way communication path between a sending process and a receiving process. In an external CICS interface implementation, each pipe maps onto one MRO session, where the client program represents the sending process and the CICS server region represents the receiving process.

5. Close_Pipe
6. Deallocate_Pipe

The processing of an EXCI CALL-level command is shown in Figure 47.

The EXEC CICS interface

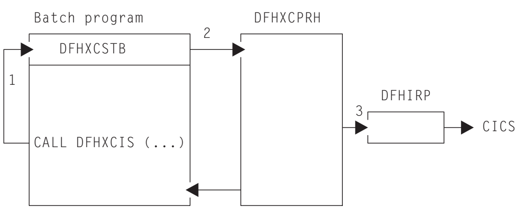
The external CICS interface provides a single, composite command–EXEC CICS LINK PROGRAM– that performs all six commands of the EXCI CALL interface in one invocation. The processing of an EXEC CICS LINK command is shown in Figure 48 on page 159.

This command takes the same form as the distributed program link command of the CICS command-level application programming interface.

API restrictions for server programs

A CICS server program invoked by an external CICS interface request is restricted to the DPL subset of the CICS application programming interface. This subset (the DPL subset) of the API commands is the same as for a CICS-to-CICS server program.

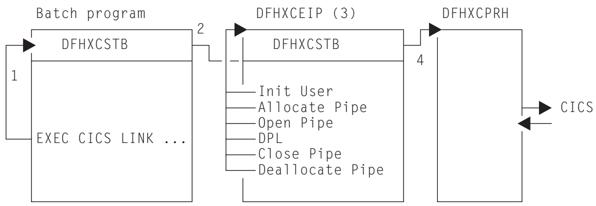
For details about the DPL subset for server programs, see the *CICS Application Programming Guide*.



Notes:

1. An EXCI CALL API request is issued, and invokes the DFHXCIS entry point in the EXCI stub, DFHXCSTB.
2. DFHXCSTB locates DFHXCPRH, and invokes it to process the EXCI request. If DFHXCPRH is not found, DFHXCSTB loads DFHXCPRH before invoking it.
3. DFHXCPRH sets up the control blocks needed for the EXCI request. For a DPL request, DFHXCPRH invokes DFHIRP to pass control to CICS.

Figure 47. External CICS interface, CALL-level API



Notes:

1. An EXCI EXEC API request is issued, and invokes the DFHXCEI entry point in the EXCI stub, DFHXCSTB.
2. DFHXCSTB locates DFHXCEIP, and invokes it to process the EXCI request. If DFHXCEIP is not found, DFHXCSTB loads DFHXCEIP before invoking it.
3. DFHXCEIP converts the EXCI EXEC-level request into a series of EXCI CALL-level requests.
4. The CALL-level requests result in calls to the EXCI stub, DFHXCSTB (as in Figure 47 on page 158).

Figure 48. External CICS interface, EXEC-level API

Modules

Module	Function
DFHXCALL	EXEC-level API macro. Invoked by the CICS translator when processing EXCI EXEC-level requests.
DFHXCDMP	dump services. Calls the CICS SVC to issue SDUMP macro requests, to take an SDUMP of the EXCI address space.
DFHXCSTB	stub link-edited with applications that want to use EXCI.
DFHXCEIP	EXEC-level API handler. The main EXCI module that processes EXCI EXEC-level requests.
DFHXCO	options macro for generating the DFHXCOPT options table.
DFHXCOPT	options table to customize the EXCI environment.
DFHXCPLD	Assembler-language parameter list definitions. Copybook defining the parameters for use with the EXCI APIs.
DFHXCPLH	C parameter list definitions. Copybook defining the parameters for use with the EXCI APIs.
DFHXCPLL	PL/I parameter list definitions. Copybook defining the parameters for use with the EXCI APIs.
DFHXCPLO	COBOL parameter list definitions. Copybook defining the parameters for use with the EXCI APIs.
DFHXCPRH	program request handler The main EXCI module that processes EXCI CALL-level requests.
DFHXCRCD	Assembler-language return code definitions. Copybook defining the return codes for use with the EXCI APIs.
DFHXCRCH	C return code definitions. Copybook defining the return codes for use with the EXCI APIs.
DFHXCRCL	PL/I return code definitions. Copybook defining the return codes for use with the EXCI APIs.
DFHXCRCO	COBOL return code definitions. Copybook defining the return codes for use with the EXCI APIs.
DFHXCSVC	SVC services. Invoked by the CICS SVC to issue an SDUMP macro to take an SDUMP of the EXCI address space.
DFHXCTAB	language table. Copybook defining the syntax of the EXCI EXEC language for use by the CICS translator.

Module	Function
DFHXCTRA	global trap program. The EXCI equivalent of the DFHTRAP module, providing the service with ability to collect extra diagnostic information.
DFHXCTRD	local trap parameter list definition. Defines the parameter list passed to DFHXCTRA and all EXCI trace points used by DFHXCTRA.
DFHXCTRP	trace services. Writes EXCI trace entries to the EXCI internal trace table.
DFHXCTRI	trace initialization. Initializes EXCI trace services.
DFHXCURM	User-replaceable program that allows the user to modify the applid of the CICS region to which the EXCI request is to be issued.

Exits

There are no exit points for the EXCI.

Trace

The EXCI has its own internal trace table in the EXCI address space where the client program is running. EXCI trace entries can also be written to the MVS GTF trace data set.

EXCI trace point IDs are EXxxxx, with a trace level of 1, 2, or Exc.

For more information about EXCI tracing, see the *CICS External Interfaces Guide*.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 23. Field engineering program

The field engineering program (DFHFEP) is a CICS system service function primarily designed for an IBM field engineer to use when installing new terminals. When CICS is running, this program (invoked by the CSFE transaction) transmits a set of characters to the requesting terminal. In addition, the program can be used to echo a message; that is, it repeats exactly what is keyed at the terminal.

This program also supports some general debugging functions.

Design overview

When used for testing terminals, DFHFEP first prepares for device-dependent conditions. It then issues a storage control FREEMAIN, followed by a GETMAIN for storage for the ENTER message, which it writes using terminal control WRITE, READ, and WAIT macros. Finally, if **print** was requested, the character set is printed; if **end** was requested, the completion message is issued; otherwise the input is echoed.

DFHFEP performs all the requests made by the CSFE transaction. In addition to the terminal test function, CSFE can request the activation or deactivation of:
- System spooling interface trace
- Terminal builder trace
- Storage freeze
- Storage violation trap
- Global trap/trace exit.

See the *CICS Supplied Transactions* manual for details of the command syntax and functions provided.

Modules

DFHFEP

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

Chapter 24. File control

File control provides a facility for accessing data sets, files, and data tables, using keyed or relative-byte-address (RBA) access through the virtual storage access method (VSAM), the basic direct access method (BDAM), shared data table services and the coupling facility data tables server. VSAM data sets can be accessed in either RLS or non-RLS mode. RLS mode allows sharing of data sets across a parallel sysplex. File control allows updates, additions, deletions, random retrieval, and sequential retrieval (browsing) of logical data in the data sets. If VSAM is used, access to logical data can be via a VSAM alternate index path, as well as through the base data set.

File control reads from, and writes to, user-defined data sets and data tables, gathers statistics, and acquires dynamic storage for I/O operations. File control uses control information defined by the user in the file control table (FCT). This table describes the physical characteristics of all the data sets, and any logical relationships that may exist between them.

Design overview

File control provides the following services and features:
- Random record retrieval
- Random record update
- Random record addition
- Random record deletion (VSAM only)
- Sequential record retrieval
- BDAM deblocking
- Enabling and disabling of files, making them accessible to applications
- Opening and closing of files for the access method
- Exclusive control of records during update operations
- Mass record insertion (VSAM only)
- Automatic journaling and logging.

Deblocking services for BDAM data sets

CICS provides deblocking of logical records on a direct-access (BDAM) data set. This service is provided for both fixed-length and variable-length records. The data set must have been created according to standard System/370 record-formatting conventions.

Concurrency control

Protection is provided against the concurrent updating (adding, deleting or changing) of a data set record by two or more transactions (or strictly speaking, two or more units of work; a transaction may optionally consist of a sequence of units of work). This protection is in most cases achieved using locking. If a second unit of work attempts to update a record which has been locked by another unit of work, the second unit of work is normally queued until the first releases its lock. If the lock has been converted into a retained lock (this is done if a syncpoint failure occurs) then the second unit of work gets an error response rather than being queued. An optimized alternative to locking is used to achieve concurrency control for coupling facility data tables. This is described in the section 'Concurrency control for coupling facility data tables'.

For a VSAM data set being accessed in non-RLS mode, CICS acquires locks (or enqueues) using the NQ domain that prevent the same record from being updated by more than one unit of work at a time. If the file is recoverable, then the lock is not released until syncpoint (that is, the end of the unit of work), otherwise it is released when the request thread completes. A request thread consists, for example, of a read update followed by a rewrite. In non-RLS mode, VSAM also provides a form of concurrency control known as **exclusive control**. The sphere of exclusive control is the control interval (CI), and this means that two different records cannot be concurrently updated if they are both within the same CI. Exclusive control is only maintained while a record is being updated, and is released as soon as the operation is complete.

For a VSAM data set being accessed in RLS mode, VSAM acquires locks at the record level to prevent the same record from being updated by more than one unit of work within the sysplex at a time. If the data set is recoverable, then the lock is not released until syncpoint, otherwise it is released when the request sequence completes. There is no CI locking with RLS mode.

For a recoverable BDAM file, CICS acquires locks using the NQ domain that prevent the same record from being updated by more than one unit of work at a time.

Concurrency control for coupling facility data tables

Concurrency control for coupling facility data tables is provided by using one of two update models provided by coupling facility data tables support (CFDT support).

The default is the locking update model, in which the CFDT server acquires locks at the record level to prevent the same record from being updated by more than one unit of work within the sysplex at a time. If the data set is recoverable, then the lock is not released until syncpoint, otherwise it is released when the request sequence completes.

The contention update model is an optimized alternative to using locking to achieve update integrity (concurrency control). With this model, which can be specified on a per-data table basis, no locks are acquired when a record is read for update, but if another unit of work subsequently changes or deletes this record, then the first unit of work will be informed that the record has changed (or been deleted) when it comes to rewrite or delete the record itself. The occurrence of such a contention is detected by the CFDT server, and the contention update model is only available for coupling facility data tables.

Sequential retrieval

A facility supported by CICS file control is the sequential retrieval of records from the database. This facility is known as browsing. To initiate a browse operation, the user provides either a specific or generic (partial) record reference (key) for the point at which sequential retrieval is to begin. Each subsequent get request by the user initiates retrieval of the next sequential record. The application, while in browse mode, can issue random get for update requests to a different data set, without interrupting the browse operation. For VSAM files accessed in RLS mode, the application can update the records that it is browsing. For VSAM files accessed in non-RLS mode, and BDAM files, in order to update a record of the same data set, the application must first terminate the browse operation. The same application can concurrently browse several different data sets and browse the same data set with multiple tasks.

With VSAM data sets, the application can skip forward during a browse operation to bypass unwanted data.

All types of CICS data tables (CICS-maintained, user-maintained and coupling facility) can be browsed.

Read Integrity

When a file is accessed in RLS mode, three levels of read integrity are supported:

- UNCOMMITTED read integrity is the same level of read integrity as is supported for non-RLS requests. With this level of read integrity, read requests can return data which has not yet been committed, and which might subsequently be backed out.
- CONSISTENT read integrity. With this level of read integrity, read requests are serialized with concurrent update activity for the record, so that a read request will wait until data which is being updated has been committed (or until the update has completed, for a non-recoverable data set). This means that read requests will always see commit-consistent data.

- REPEATABLE read integrity. With this level of read integrity, additional locking is used so that in addition to waiting for updates to be committed, records that have been read within a unit of work cannot be updated until the unit of work completes. This means that if a read is repeated within a unit of work, the same data will be returned.

Backout logging

File control will perform automatic logging of file operations which update recoverable files. This logging is written to the CICS system log stream. In the event of either a system or a transaction failure, the information can subsequently be used to restore the recoverable data set as though the current transaction had never run.

For coupling facility data tables, the CFDT server performs its own logging, and is responsible for backing out updates in the event of a failure.

Forward Recovery Logging

If a file (non-RLS VSAM) or data set (RLS or non-RLS VSAM) is defined to be forward recoverable, then CICS will perform automatic logging of file operations which update it. This logging is written to the forward recovery log stream specified on the file definition or data set. In the event of a failure, the information can be used to forward recover from a backup copy of the data set.

Forward recovery support is not provided for user-maintained data tables or coupling facility data tables.

Automatic journaling and logging

Except in the case of user-maintained data tables and coupling facility data tables, CICS provides optional automatic journaling and logging facilities for records that are updated, deleted from, or added to a file control data set. Automatic journaling is specified in the file control table, by the user, for each data set affected. For a specified data set, a record read for update, a new record added, or an existing record deleted is automatically written to the specified journal. To allow journaled records to be associated with the appropriate data set (instead of with the CICS file name), a special record is journaled showing the current data set allocation whenever it changes.

Use of concurrent tasks

The file control non-RLS VSAM interface program (DFHFCVR) uses a change-mode request to the dispatcher to allow VSAM I/O requests and VSAM UPAD exit code to run under a concurrent task. This provides overlapping of processing in a multiprocessor environment.

RLS requests use a different mechanism: SMSVSAM assigns each request its own SRB, allowing MVS to concurrently schedule requests in an analogous way to that provided by subtasking for non-RLS.

Shared Data table services

Shared data tables (that is, CICS-maintained and user-maintained data tables) are managed by a set of OCO modules, referred to in this book as “data table services”. The services are invoked by a branch-and-link interface passing a parameter block.

Services provided include the following:
- Initialization
- Open, close, and load of tables
- Retrieval and update of table records
- Backout and commit of table changes
- Statistics.

For files that are defined by the user as CICS-maintained or user-maintained data tables, file control invokes these services at appropriate points in the processing of application requests.

Coupling facility data tables server

Coupling facility data tables are managed by a OCO modules within the CICS address space, along with a separate address space, referred to as the "Coupling Facility Data Tables Server". The CFDT server provides access to coupling facility data tables residing in a coupling facility data tables pool, so that they can be shared by CICS regions across a parallel sysplex. Refer to the *CICS Supplied Transactions* for more details about CFDT servers.

For files that are defined by the user as accessing coupling facility data tables, file control makes calls to the CFDT server at appropriate points in the processing of application requests.

How CICS processes file control requests

CICS receives file control requests from applications through the EXEC interface. This section describes only the mainstream processing for such requests. It does not describe exceptional conditions. For guidance about exceptional conditions, see the *CICS Application Programming Guide*. For general-use programming interface information about exceptional conditions, see the *CICS Application Programming Reference* manual. This section also does not provide details about the specific processing for requests to any kind of data table.

Processing using VSAM

For VSAM data sets, this section describes the processing followed when the file is being accessed in non-RLS mode. For RLS mode, the processing is broadly similar, although it differs in some of the interfaces used to VSAM, and the locking mechanisms are very different.

Note: File control processing is constrained by the availability of buffers, CICS strings and (for local shared resource (LSR) files) LSR strings. Tasks can get suspended during the execution of any file control request if there are not enough strings or buffers available for the immediate processing that is to be done.

With VSAM RLS, a task waiting for buffers will be suspended in VSAM rather than in CICS.

Processing using Data Tables

For shared data tables (CICS-maintained and user-maintained data tables), processing is broadly similar to that for non-RLS VSAM. The main differences are that, for remote files, non-update requests may be processed locally instead of being function shipped, and that, in cases where a request cannot be satisfied from a data table, it may be converted into a non-RLS or RLS VSAM request to be processed by DFHFCVS or DFHFCRS, or function shipped via DFHFCDTX.

For coupling facility data tables, processing is also broadly similar to that for non-RLS VSAM. The main difference is that instead of issuing the request to VSAM, a call or calls are made to entry points within the CFDT server, which then processes the request and returns the results. A task accessing a coupling facility data table may occasionally be suspended in the CFDT server.

Note that the following processing sections do not describe data table processing explicitly.

General request processing

All file requests, whatever the request and whatever the file access method, follow the same general sequence of steps:

1. User exit XFCREQ is called.
2. The request is converted from EXEC parameter list form to FCFR interface form.
3. If this is the first file access request by the transaction, a FRAB is obtained and its address stored in Recovery Manager's FC Token. The FRAB provides the anchor for file request state for this transaction.

4. If this is the first request to this file by the transaction, a FLAB is obtained and the file control table entry is located. If the file is remote or an explicit SYSID has been specified on the request, the FLAB is marked with a remote indicator. If this is not the first request to the file, then the FLAB is located that repressents accesses made to the file by this transaction.
5. If this is the first, or only, request of a request sequence, a FRTE is obtained. If this is not the first request in a request sequence, the FRTE that represents the sequence is located. rather than being function shipped.
6. If the request is to a local file, and if resource security is active, the security check is made, unless a check has already been made within the current UOW for this file.
7. If the request is to a local file and the file is not already open, it is opened and its access method dependent attributes are saved in its file control table entry.
8. The SERVREQ attributes of the file are checked.
9. For READ and browse requests, SET storage is released and/or obtained, as necesssary.
10. The access method specific request processor is called as follows:
    - DFHFCVS for non-RLS VSAM files
    - DFHFCRS for RLS VSAM files
    - DFHFCBD for BDAM files
    - DFHFCDR for coupling facility data tables
    - DFHFCDTS for user-maintained data tables
    - DFHFCDTS for non-update requests to CICS maintained data tables
    - DFHFCVS for update requests to CICS maintained data tables
    - DFHFCRF for requests that are to be shipped to a remote region
11. CICS has checked whether the file is defined as local or remote. If it is remote, the request is function-shipped to the file-owning region, where CICS processes the request as if it had originated locally.

    There is an exception for CICS-maintained and user-maintained data tables, for which non-update requests are treated as local rather than being function shipped.

    Note that RLS support and coupling facility data tables support both provided shared access within a parallel sysplex without the use of function shipping. Files which use either of these types of sharing will be defined as local on all systems which wish to share the data set (in the case of RLS support) or data table (CFDT support).
12. SET storage is obtained for BDAM files or below the line READ requests.
13. The FRTE is released if the request sequence has ended and the file is closed if a close is pending, this FRTE is the last user and the FLAB indicates that the file can be closed.
14. The FCFR responsed are converted to EXEC parameter list responses. In particular, the EIBRCODE and RESP2 values are constructed.
15. User exit XFCREQC is called.

READ request processing

The course of READ request processing depends on the access method, and whether or not the UPDATE option is specified on the request:

VSAM processing:

1. The supplied keylength is validated.
2. A VSAM work area (VSWA) is created. This includes the request parameter list (RPL) that will be passed to VSAM.

   The processing that follows depends on whether the UPDATE option was specified on the READ request.

   UPDATE option not specified:

   a. The RPL is completed, and a call made to VSAM to get the record.

b. If the request specifies INTO and the record is too large for the user-specified area, the request is reissued specifying a work area large enough to hold the record. The record is then copied to the user-specified area in truncated form, and the LENGERR condition is raised.
c. The VSWA is freed.
d. The read is journaled if specified in the FCT entry.

UPDATE option specified:

a. The UPDATE flag is set in the RPL.
b. An attempt is made to read the record by issuing the VSAM request. READ UPDATE requires exclusive control of the control interval (CI) containing the record. VSAM manages the locking mechanism for control intervals. If the CI is already locked, VSAM returns an error and the requesting task is forced to wait on resource type FCXCWAIT.
c. CICS file control acquires a record lock on the record just read, using a CICS ENQUEUE request. The record lock prevents any other transaction from updating the record before the owning transaction has reached a syncpoint (for recoverable files), or before the REWRITE, DELETE, UNLOCK or syncpoint that completes the request sequence (non-recoverable files).
d. Exclusive control of the CI is retained until the REWRITE, DELETE, or UNLOCK request that follows the READ UPDATE has been completed, or until the next syncpoint.

   The CICS record lock (if any) is retained until the next syncpoint, in case the transaction updating the record abends and dynamic transaction backout processing is necessary.
e. If the file is recoverable the request is logged. If required, the request is also recorded in a user-specified journal.

BDAM processing:

a. A file I/O area (FIOA) is obtained.
b. If the UPDATE option has been specified:
   1) The address of the RIDFLD is saved in the FIOA.
   2) If the data set is recoverable, the RIDFLD is ENQUEUEd on to lock the record against other updates. The ENQUEUE is retained until the next syncpoint.
c. The KEYLENGTH is checked for validity.
d. The key field is converted from character string format (TTTTTTRR) to binary format (TTR), if necessary.
e. A BDAM READ request is issued. If the READ is successful, the required block is returned in the FIOA.
f. The key field returned by BDAM is converted from binary format to character string format, if necessary.
g. If the file is recoverable and UPDATE has been specified, the request is logged. If required, the request is also recorded in a user-specified journal.
h. If deblocking is required, the required record is located in the block that has been returned by BDAM:
   1) If DEBREC has been specified, the record number is used to locate the record.
   2) If DEBKEY has been specified, the embedded key is used to locate the record.

WRITE request processing

The course of WRITE request processing depends on the access method, and for VSAM access on whether the file is a data table: **VSAM processing:**

1. The KEYLENGTH is checked for validity. If it is incorrect, the INVREQ condition is raised.
2. A VSAM work area (VSWA) is created. This includes the request parameter list (RPL) that will be passed to VSAM.

   Different paths are now followed depending on the type of file.

   ESDS file:

a. If the file is recoverable or writes are to be journaled then
   1) If this is not the first write of a sequence and the ESDS write lock is being waited for by another transaction, then release the lock and end this sequence, logging the completion if recoverable.
   2) If this is (or has become) the first write of a sequence, acquire the ESDS write lock for the data set.
b. If the file is recoverable, the WRITE ADD request is recorded in the CICS system log.
c. If required, the WRITE ADD request is recorded in a user-specified journal.
d. Any fields in the RPL not supplied when the VSWA was created are completed.
e. The RPL is set to point to the user-specified data area. If the user specified a record that is too large for the file, the length in the RPL is set to the maximum length, so that the record is truncated.
f. A VSAM PUT request is issued to write the record.
g. If the file is recoverable, a CICS record lock is obtained for the record that has just been written. The record lock will be retained until the next syncpoint, in case the transaction writing the record abends and dynamic transaction backout processing has to be performed.
h. If the file is recoverable, the after-image of the record is logged for forward recovery and a write complete record is written on the system log.
i. If not a MASSINSERT the ESDS write lock is released, if held.

KSDS or RRDS file:

a. For KSDS requests, the RIDFLD key specified in the request is checked against the key field in the record to be written. (The record is currently in the application FROM data area.) If it does not match, the INVREQ condition is raised.
b. If the file is recoverable and not in load mode:
   1) A CICS lock is obtained on the record that is to be written, and an attempt is made to read the record (by means of a VSAM GET request) to discover whether it already exists in the file. If it does, the DUPREQ condition will be raised on the write to VSAM.
   2) If the file is a KSDS, and if this request is part of a MASSINSERT, or if a MASSINSERT is in progress, the read is issued with GTEQ to find the next record in the base data set. A lock is created, using the key of this next record, to prevent other transactions from inserting records into the empty range.
   3) If there is no existing record with the given key, the WRITE ADD request to VSAM is recorded in the CICS system log and, if required, in a user-specified journal.
c. If the file is not recoverable or in load mode, the WRITE request is recorded, if required, in the user-specified journal, and if recoverable a record lock is obtained and the write logged.
d. Any fields in the RPL not supplied when the VSWA was created are completed.
e. If a data table is associated with the base cluster (the data table will be a CICS-maintained table, as user-maintained and coupling facility data tables follow a separate processing path which is not described here). a data table pre-add is issued to place the record in the table as a not-yet-valid entry. If the file is recoverable, a record lock is already held; if not, a lock is acquired before the data table service is called.
f. A VSAM request is issued to write the record.
g. If the file is recoverable, the after-image of the record is logged for forward recovery.
h. If required, the after-image is recorded in a user-specified journal.
i. If the file is a data table, a data table request is issued to complete the add to the data table by validating the record. If a record lock was obtained for a non-recoverable file, it is released.

3. If the MASSINSERT option has *not* been specified on the WRITE request, the VSWA for the operation is released.

If MASSINSERT has been specified, the VSWA is not released, because it is likely to be needed for subsequent WRITE operations. In this case, the end of MASSINSERT processing is notified to VSAM by the CICS UNLOCK function. (See “UNLOCK request processing” on page 171.)

Specifying MASSINSERT causes exclusive control of the CI to be acquired. Exclusive control is released by issuing an UNLOCK request. To avoid deadlocks, this should be done immediately after the last WRITE MASSINSERT request.

BDAM processing:

1. The KEYLENGTH is checked for validity. If it is incorrect, the INVREQ condition is raised.
2. The WRITE command input is checked to ensure that MASSINSERT has not been specified—BDAM does not support MASSINSERT processing. If it has, condition INVREQ is raised.
3. A file I/O area (FIOA) is obtained.
4. If the file is recoverable, the record to be written is ENQUEUEd on. The lock is retained until the next syncpoint.
5. The record to be written is copied from the user-supplied data area to the FIOA. If the record is too large, it is truncated.
6. If the file is recoverable, the request is logged. If required, the request is also recorded in a user-specified journal.
7. The key field is converted from character string format to binary format, if necessary, and the BDAM I/O request issued.
8. The key returned by BDAM is converted from binary format to character string format, if necessary, and passed to the application.
9. A supervisor call (SVC 53) is issued to release BDAM exclusive control, if necessary.
10. The FIOA is FREEMAINed.

REWRITE request processing

The REWRITE request is used to write a record back to a file following a READ UPDATE request. **VSAM processing:**

1. The RPL is set to point to the user-specified data area. If the user specified a record that is too large for the file, the length in the RPL is set to the maximum length, so that the record is truncated.
2. The RPL is completed.
3. If there is a data table associated with the base cluster (this will be a CICS-maintained table, as user-maintained tables follow data table processing):
   a. If the file is nonrecoverable, a record lock is obtained. (If the file is recoverable, a lock is already held).
   b. A data table request is issued to invalidate the record in the table before the VSAM update.
4. VSAM is called to PUT(UPDATE) the record. Exclusive control of the CI, which was obtained for the preceding READ UPDATE request, is released, but the CICS record lock (for recoverable files) is retained until the next syncpoint, in case the transaction abends and dynamic transaction backout processing is necessary.
5. If there is a data table associated with the data set, the table record is updated and its validity is reinstated, by issuing a call to data table services. If the file is nonrecoverable, the record lock is released.
6. If the file is recoverable, and if the record is successfully rewritten, the after-image is written to the log for forward recovery.
7. The VSWA for the operation is released.

Note: When a record is updated by way of a path, the corresponding alternate index is updated by VSAM to reflect the change. However, if the record is updated directly by way of the base, or by a different path, the AIX® will only be updated by VSAM if it has been defined to VSAM (when created) to belong to the **upgrade set** of the base data set.

BDAM processing:

1. The FIOA that was used in the corresponding READ UPDATE request is located, and the modified record read into it from the user-specified area. If the record is too long, it is truncated.
2. A FREEMAIN call is issued to release the FWA.
3. If the file is recoverable, the request is logged. If required, the request is also recorded in a user-specified journal.
4. The key field is converted from character string format to binary format, if necessary, and the BDAM I/O request issued.
5. The key returned by BDAM is converted from binary format to character string format, if necessary, and passed to the application.
6. A supervisor call (SVC 53) is issued to release BDAM exclusive control, if necessary.
7. A FREEMAIN call is issued to release the FIOA.

UNLOCK request processing

The UNLOCK request is used to release exclusive control obtained during a READ UPDATE (VSAM or BDAM) or WRITE MASSINSERT (VSAM only) request.

VSAM processing (including CICS-maintained data tables):

1. The VSWA for the operation is released, together with associated storage.
2. An ENDREQ request is sent to VSAM. This releases exclusive control of the CI, if it is held, and frees any VSAM strings.

BDAM processing:

1. A supervisor call (SVC 53) is issued to release BDAM exclusive control, if necessary.
2. A FREEMAIN call is issued to release the FIOA.

DELETE request processing

The course of DELETE request processing depends on whether a RIDFLD has been specified. The processing for user-maintained data tables differs from that for CICS-maintained data tables. DELETE requests are not valid for VSAM ESDS or for BDAM files.

VSAM processing (including CICS-maintained data tables):

1. If a RIDFLD has been specified:
   a. If a KEYLENGTH has been specified, it is checked for validity.
   b. If the GENERIC option has been specified, and the file is *not* a KSDS, condition INVREQ is raised.
   c. A VSWA is created.
2. If no RIDFLD was specified, the SERVREQ attribute of the file is checked to ensure that DELETE requests are valid for this file. If not, the INVREQ condition is raised.

   If a RIDFLD has been specified, the cycle of actions described below is performed once if GENERIC has not been specified, or is repeated until there are no more records containing the generic key, if GENERIC has been specified.

   Start of cycle:
3. VSAM is requested to GET for UPDATE a record with the specific or generic key. GET UPDATE processing requires exclusive control of the CI. The record is read into an internal buffer.

   The generic key value, if supplied, is checked against the key contained in the record. If it does not match, there are no more records containing the generic key in the file.
4. If the file is recoverable:
   a. A CICS record lock is obtained for the record. This will be held until the next syncpoint.
   b. The VSAM GET UPDATE request is recorded synchronously on the system log.

   c. A CICS range lock is obtained for the record to be deleted if a MASSINSERT is in progress. This is to prevent an end-of-range record from being deleted while the range is in use for a MASSINSERT sequence.
5. If there is a data table (which will be CICS-maintained) associated with the base cluster, a record lock is acquired if the file is nonrecoverable, and a data table pre-update call is issued to invalidate the record before the VSAM update.
6. A VSAM ERASE request is issued, to delete the record from the file.
7. If there is a data table associated with the base cluster, the record is deleted from the table by issuing a call to data table services. If the file is nonrecoverable, the record lock is released.
8. If a range lock was acquired, it is released.
9. If the file is recoverable, a WRITE DELETE record is written in the system log for forward recovery.
10. If required, a WRITE DELETE record is written to a user-specified journal.

    End of cycle.
11. The VSWA is released.

STARTBR and RESETBR request processing

STARTBR and RESETBR request processing are very similar, and are described together.

VSAM processing:

1. A VSWA is created if STARTBR.
2. The user key is recorded in the VSWA for use in subsequent BROWSE processing.
3. A call is made to VSAM to point to the record, and to acquire shared control of the CI.

BDAM processing:

1. An FIOA is obtained and initialized if STARTBR.
2. The initial key is saved in the FIOA, converting the key from character string format to binary format if necessary.
3. If deblocking is required, the deblocking indicator (DEBREC or DEBKEY) is saved in the FIOA.

READNEXT and READPREV request processing

READNEXT and READPREV request processing are very similar, and are described together.

VSAM processing:

1. A check is made that READPREV with a generic key was not requested. If it was, condition INVREQ is raised.
2. If KEYLENGTH was specified, it is checked for validity. If it is incorrect, the INVREQ condition is raised.
3. The RPL options are set.
4. If SET is specified, an internal work area is obtained and the RPL is set to point to the work area. The area is either above or below the 16MB line, depending on the requirements of the application.
5. If INTO is specified, the RPL is set to point to the user-specified area.
6. A VSAM request is issued to read the record. Shared control of the CI is needed, and the request will not succeed if some other task already has exclusive control. In such a case, a call is made to VSAM to reestablish the correct position in the file. The task then waits until VSAM informs CICS that the CI is available to the task. CICS resumes the task, which can now acquire shared control and obtain the required record.
7. If SET is specified, the SET pointer points to the work area.
8. If INTO is specified, a check is made to see if the record is too large to fit into the user-specified area. If it is too large, the request is reissued using an internal work area, the data is copied from the work area into the user-specified area and truncated, and the LENGERR condition is raised.
9. If required, the request is recorded in a user-specified journal.

BDAM processing—READNEXT requests:

1. A check is made that READPREV was not issued. If it was, condition INVREQ is raised.
2. The FIOA that was created on STARTBR is located.
3. If a new block is required, a BDAM I/O request is issued to get it.
4. If deblocking is required, the required record is located in the block that has been returned by BDAM:
   a. If DEBREC has been specified, the record number is used to locate the record.
   b. If DEBKEY has been specified, the embedded key is used to locate the record.
5. If INTO is specified, the record or block is moved from the FIOA to the user-specified area. If the record is longer than the user-specified area, it is truncated, and the LENGERR condition is raised.
6. If SET is specified, the SET pointer points to the record in the FIOA.
7. The RIDFLD of the record is returned to the application.
8. The current browse position is recorded in the FIOA.

ENDBR request processing

The ENDBR request is used to end a browse session on a file. To avoid deadlocks, ENDBR must be issued when the browse session is complete.

VSAM processing:

1. An ENDREQ request is sent to VSAM. This frees any VSAM strings that are held, and relinquishes shared control of the CI.
2. The VSWA for the operation is released.

BDAM processing:

- The FIOA that was used for the browse session is FREEMAINed.

Control blocks

Figure 49 on page 174 shows the major control blocks associated with file control. Control blocks which are not shown in this diagram include those relating to coupling facility data tables support.

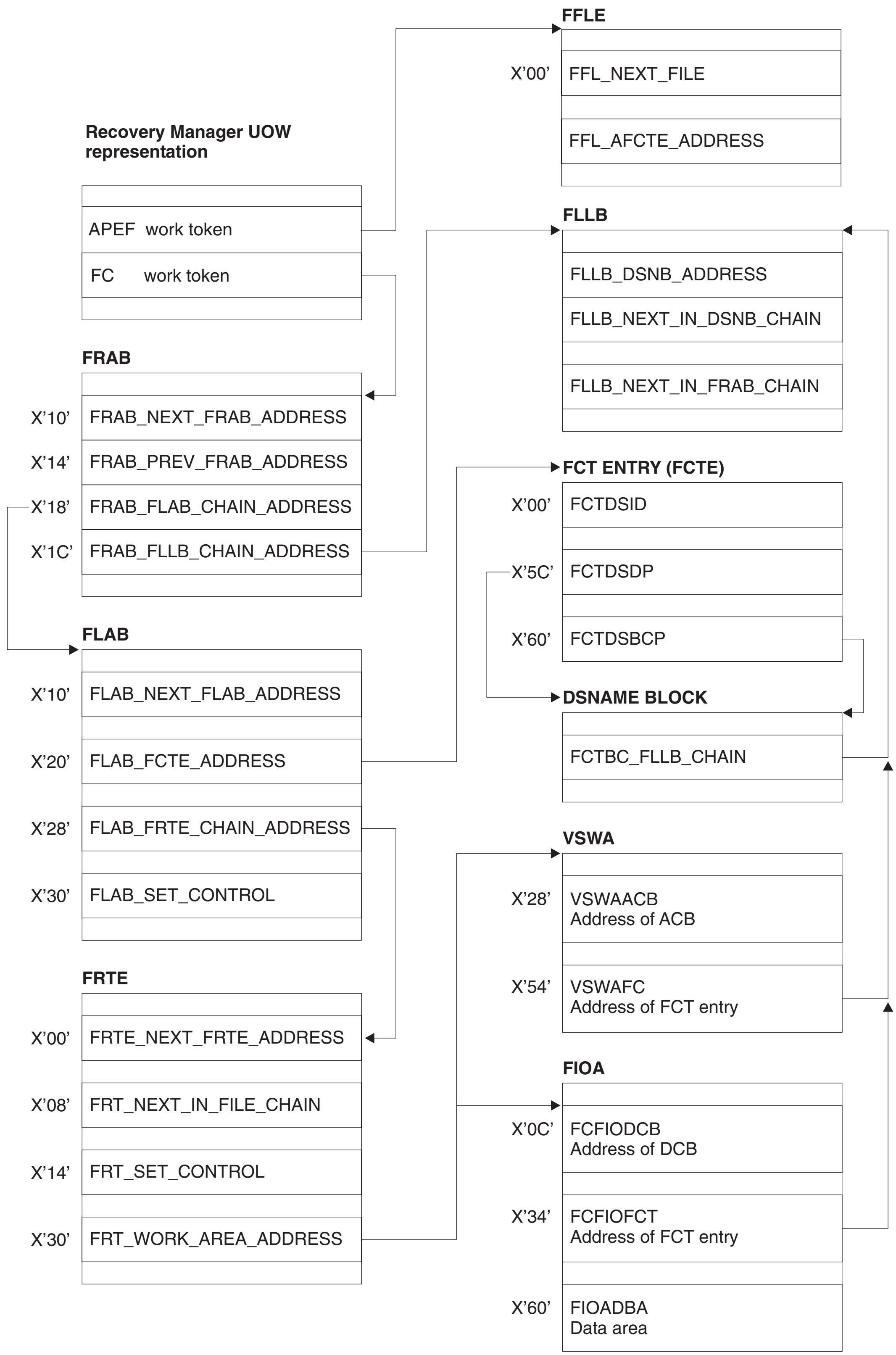


Figure 49. Control blocks associated with file control (Part 1 of 2)

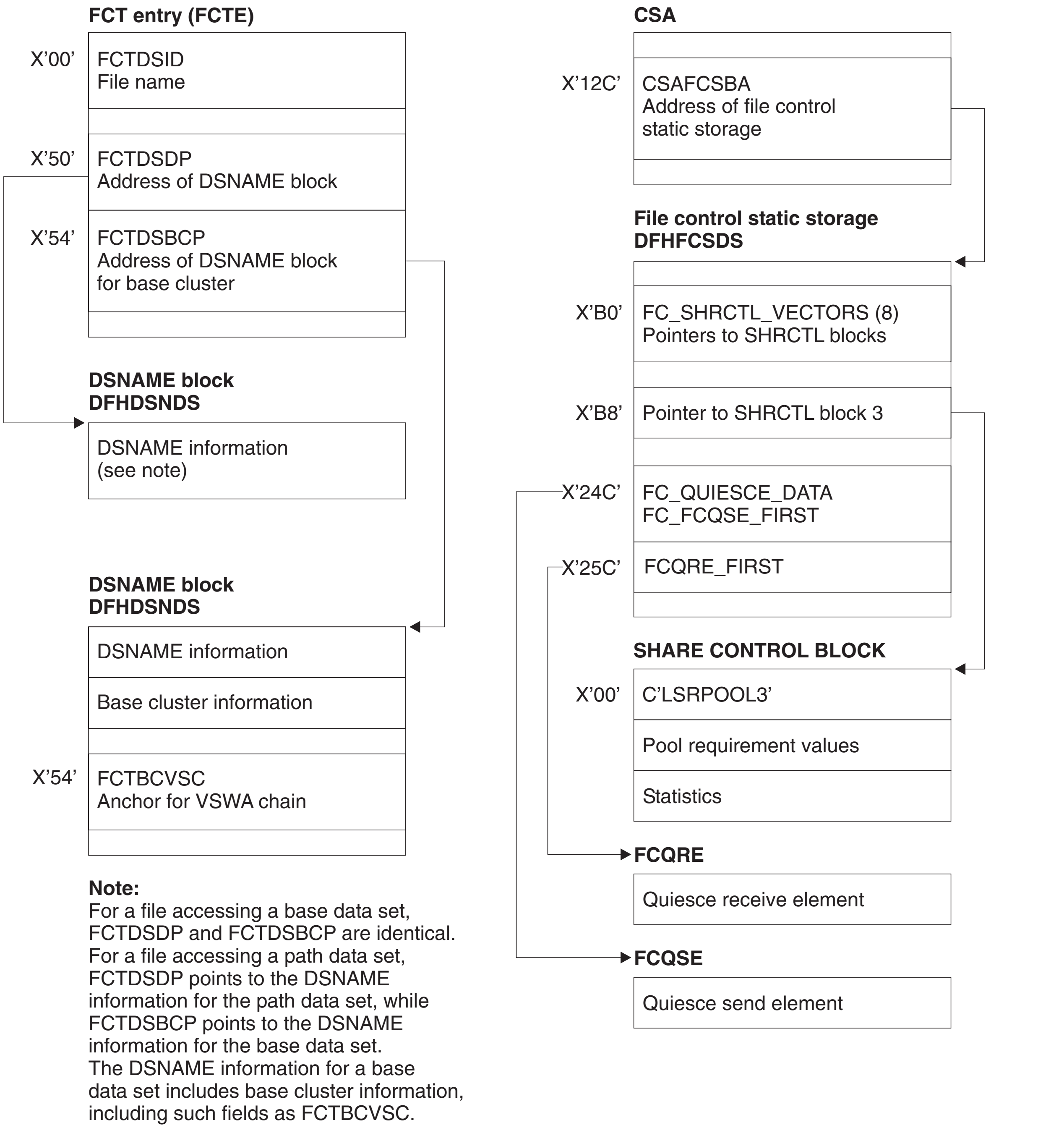


Figure 49. Control blocks associated with file control (Part 2 of 2)

Note: The pointer to the DSNAME block, FCTDSDP, is different from the pointer to the base cluster DSNAME block, FCTDSBCP, only when the FCT entry does not represent a base. DSNAME blocks that do not correspond to bases do not have the base cluster information, although the space is allocated.

These control blocks are described in the Sections "Access method control block (ACB)" on page 176 through "VSAM work area (VSWA)" on page 183.

Access method control block (ACB)

The ACB identifies to VSAM the file associated with this VSAM request. It is passed to VSAM by DFHFCRV, for RLS, or DFHFCVR, for non-RLS (it is actually the RPL, which points to the ACB, that is passed) to initiate a VSAM request. The ACB lasts as long as the associated CICS file is open; that is, it is created at file open time and deleted at file close time by DFHFCN for non-RLS or DFHFCRO for RLS.

The ACB is addressable through a pointer in the associated FCT entry. In addition, a 4-byte field appended (by CICS) to the ACB structure points back to this FCTE.

Note that the ACB is a VSAM control block.

At open time, storage is obtained from a subpool above the 16MB line. A VSAM GENCB macro is issued to generate the ACB with attributes obtained from the FCT entry. At open time, VSAM fills in more information in the ACB. Some of this is subsequently copied back into the FCTE.

The storage for the ACB is freed when the file is closed.

There is one ACB per VSAM FCT entry.

The layout of the ACB is defined by the VSAM IFGACB structure, and also by a DSECT of the same name.

ACBs are not cataloged and are not restored across WARM or emergency starts. The ACB is rebuilt every time a CICS file is opened.

A special type of ACB, known as a base cluster ACB, is created by DFHFCM to allow for the implicit opening of a base cluster, when required by a non-RLS file access through an alternate index path. In this case, the 4-byte field appended to the ACB structure points to the associated DSNAME block for the base cluster.

A second special type of ACB, known as a **control ACB** is required for VSAM RLS processing. Storage for the control ACB is obtained by DFHFCCA and filled in using the GENCB macro before registering the control ACB. The storage is freed when the control ACB is unregistered by DFHFCCA. The control ACB is passed to VSAM on calls issued by DFHFCCA. It is used for all requests that are not associated with a specific file.

Data control block (DCB)

The DCB identifies to BDAM the file associated with this BDAM request. It is passed to BDAM by DFHFCBD to initiate a BDAM request, and lasts for the lifetime of the CICS run.

The DCB is addressable through a pointer in the associated FCT entry. In addition, a 4-byte field appended (by CICS) to the DCB structure points back to this FCTE.

Note that the DCB is a BDAM control block.

There is one DCB per BDAM FCT entry.

The layout of the DCB is defined by the generalized structure IHADCB. The structure is qualified with a parameter stating that a BDAM DCB is required. There is also a DSECT of the same name.

The DCB is assembled as part of the FCT. (Note that there is no RDO for BDAM files.) DFHFCRP acquires storage for the DCB below the 16MB line and copies the DCB into it (only on cold start). The DCB is cataloged and restored across a warm and emergency start. Thus, unlike an ACB, a DCB is only built once.

Data set name block (DSNB)

The DSNB represents a physical VSAM or BDAM data set that is being accessed through one or more CICS files. It is used by file control to hold information relevant to the data set and not only to the CICS file. Also, it provides a single "anchor block" to control many requests accessing this data set through many different CICS files.

After it has been created, a DSNB survives the lifetime of a CICS run unless the user deletes it by means of an EXEC CICS SET DSNAME REMOVE command or its CEMT equivalent.

The DSNB is addressable through pointers in an FCTE entry, or through DFHTMP using the 44-character name as a key, or using the DSNB number as a key.

A DSNB is created, if it does not exist already, when an FCTE attempts to connect itself to a DSNB. This happens at file open time, or when an EXEC CICS SET FILE DSNAME command (or its CEMT equivalent) is executed.

A DSNB that represents a VSAM base data set has a **base cluster block** embedded in it, which has information specific to the base data set. Note that a BDAM data set has a small amount of information held in the base cluster block.

A DSNB representing a VSAM path has a blank base cluster block embedded in it.

Information about the base data set is obtained from the VSAM catalog when a CICS file (path or base) referencing that data set is opened. The information is stored in the base cluster block.

DSNBs are cataloged in the CICS global catalog and are restored across warm and emergency starts.

DSNBs reside above the 16MB line.

The layout of the DSNB is defined by the DFHDSNPS structure, and by the DFHDSNDS DSECT (using the DFHDSND macro).

The DFHFCDN module handles DSNAME blocks (creation, deletion, FCTE-DSNB connections). DFHFCDN also provides an interface for the EXEC layer to process DSNAME blocks through the use of EXEC CICS INQUIRE or SET DSNAME, and CEMT INQUIRE or SET DSNAME. Modules within the file control component can access the DSNBs directly through pointers in the FCTE.

File browse work area (FBWA)

The FBWA maintains the state of a browse to a data table. It is used for browsing coupling facility data tables, CICS-maintained data tables, and user-maintained data tables.

An FBWA is created when the browse is started (via a STARTBR request), and is addressed by the FRT_FBWA_ADDRESS field in the FRTE. It is stored in a file control IO buffer of the appropriate size to hold the key information.

Some of the fields are specific to CICS-maintained data tables, because the source data set will sometimes be accessed during a browse of a CICS-maintained data table.

There is a variable-length portion at the end of the FBWA which contains keys, which are pointed to by fields in the fixed hang on!

part:

- CURRENT_KEY points to the first of the key fields, which is used to hold the key returned by the most recent request.

- REQUEST_KEY points to the second of the key fields, which is used to contain the key specified at the start of a browse segment (STARTBR or RESETBR).
- NEXT_KEY points to the third of the key fields, which is used for CICS-maintained data tables to handle "gaps".

File control static storage (FC static)

File control static storage is used by file control to store information for use throughout the lifetime of a CICS run; for example, SHRCTL vectors and entry points of file control modules. It is used by file control modules and by modules outside the file control component, and lasts for the lifetime of a CICS run. It is addressed by a field in the CSA named CSAFCSBA; it is created by DFHFCIN during CICS initialization before DFHFCRP gets control, and resides above the 16MB line.

FC static storage is defined by the DFHFCSPS structure and by the DFHFCSDS DSECT.

File control quiesce receive element (FCQRE)

File control uses quiesce receive elements to communicate details of quiesce requests received from SMSVSAM. There is also a permanent error FCQRE used for communicating errors. The FCQRE contains information about the data set to which the quiesce applies (or the cache for quiesce type QUICA), the type of quiesce, and (for the error FCQRE) the type of error and error data.

Each quiesce request received from SMSVSAM via the quiesce exit results in DFHFCQX, the quiesce exit module, creating an FCQRE which is passed to DFHFCQR, the quiesce receive system task module.

Storage for FCQREs is obtained from storage MVS getmained above the 16MB line.

FCQREs are chained in a one-way linked list anchored from file control static storage. The permanent error FCQRE is also anchored from file control static storage, and is added to the FCQRE chain when an error occurs.

The layout of the FCQRE is defined by the DFHFCQRE structure and the DFHFCQRE DSECT.

File control quiesce send element (FCQSE)

File control uses quiesce send elements to communicate the details of quiesce requests that are to be sent to SMSVSAM. They contain information about the task initiating the request, the data set to be quiesced, the type of quiesce requested, and the address of an ECB which is posted by SMSVSAM when the request is completed.

Each quiesce request initiated by CICS results in DFHFCQI, the quiesce initiate module, creating an FCQSE which is passed to DFHFCQS, the quiesce send module.

Storage for FCQSEs is obtained from the FC_ABOVE subpool, which resides above the 16MB line.

FCQSEs are chained in a two-way linked list anchored from fields in file control static storage.

The layout of the FCQSE is defined by the DFHFCQSE structure and the DFHFCQSE DSECT.

File control coupling facility data table pool element (FCPE)

A file control CFDT pool element represents one connection to a Coupling Facility Data Table Pool. For each CFDT pool which can be accessed by a given MVS image, there is a CFDT server running in that image which manages access to the pool.

An FCPE is created and chained to FC static when a file definition that refers to the pool is installed and there is not already a pool element for that CFDT pool. The creation of an FCPE can occur:
- when files are installed at CICS startup,

- when files are installed using CEDA,
- when a SET FILE is issued which names a CFDT pool for which there is not already a pool element.

FCPEs are getmained from the FCPE subpool which is created by DFHFCRP during File Control Initialization, and chained to the FCPE chain in FC static. The head of the FCPE chain is the field FC_FCPE_CHAIN.

FCPEs are catalogued when they are created, so that they can be restored at emergency restart.

File control coupling facility data table pool wait element (FCPW)

The file control CFDT pool wait element (FCPW) represents a task which has tried to issue a request to a coupling facility data table that resides in a particular pool, but which has to wait because there are no available request slots. Depending on the kind of request, the FCPW will represent either a 'Locking request slot' (LRS) waiter or a 'MaxReqs' waiter. A flag in the FCPW indicates what kind of wait it is.

The FCPW is created when a task goes into a MaxReqs or LRS wait. It is getmained from the pool wait element subpool, and appended to a chain of wait elements for the pool. The wait chains are anchored in the pool element (FCPE), with one FCPW for each task that is waiting. The FCPE contains head and tail fields for the chains of LRS and MaxReqs FCPWs (FCPE_FIRST_LRS_WAITER, FCPE_LAST_LRS_WAITER, FCPE_FIRST_WAITER and FCPE_LAST_WAITER). The chains are manipulated using logic which does not require any special case code for the ends of the chains, but which does mean that when the chains are empty, the head and tail fields contain a special initial value, rather than zero.

The FCPW includes:
- A pointer to the next FCPW in the chain (if no next FCPW, this contains the special initial value).
- A pointer to previous FCPW in the chain (if no previous FCPW, this contains the special initial value).
- The suspend token for the wait.
- The task token of the waiting task.
- The suspend start time.

File control table entry (FCTE)

Each entry in the file control table defines a CICS file that is defined to be the CICS view of a VSAM or BDAM data set or a data table. The FCTE is used by all modules in the file control component (but never outside), and lasts for the lifetime of a CICS run, or from when it is created by RDO to the end of the CICS run.

The FCTE is addressable through a TMP index; its layout is defined by the DFHFCTPS structure and by the DFHFCTDS DSECT; and it resides above the 16MB line.

The FCTE contains information that can be split into three broad groups:
- CICS information about the file, including statistics
- Information that is used as input to build the VSAM ACB or BDAM DCB
- Information that is returned by VSAM, both from the ACB and direct from the VSAM catalog, when the file is opened.

An FCTE can be created in two ways:
- By defining the file using the DFHFCT TYPE=FILE macro (BDAM only).
- By defining the file online using RDO while CICS is running (VSAM only).

File control table entry (FCPW)

File control coupling facility data tables UOW pool block (FCUP)

The File Control CFDT UOW Pool Block (FCUP) represents recoverable updates made within a unit of work to one or more coupling facility data tables residing in a coupling facility data table pool. An FCUP block is created when a unit of work makes its first recoverable request to a CFDT in a given pool, at the same time as an RMC link is added to represent the recoverable update.

There is one FCUP block per UOW per recoverably-updated CFDT pool. The FCUP is getmained and freemained from the FCUP subpool using the storage manager quickcell mechanism. The FCUP blocks for a unit of work are chained from the FRAB for that unit of work, addressed by FRAB_FCUP_CHAIN_ADDRESS.

An FCUP block contains:

- Forward and back pointers for the chain of FCUP blocks relating to this unit of work.
- The name of the CFDT pool.
- The CFDT RMC link token.
- A pointer to the pool element for the CFDT pool.
- A pointer back to the owning FRAB.

File input/output area (FIOA)

The FIOA is analogous to the VSWA for VSAM, in that it represents the request to BDAM. Embedded in the FIOA is what is known as the data event control block (DECB), which is passed to BDAM to initiate the request.

The FIOA is used by DFHFCBD when processing browse requests against BDAM files. It holds position in a browse when browsing a BDAM file.

An FIOA survives as long as the DECB needs to survive to complete the BDAM request; for example, it survives from READ UPDATE to the REWRITE request.

The address of the FIOA is held in the file request thread element (FRTE) in the FRT_WORK_AREA_ADDRESS field.

Storage for the FIOA is acquired from below the 16MB line.

The layout of the FIOA is defined by the DFHFIOA DSECT.

File lasting access block (FLAB)

The FLAB serves as an anchor for the set of file request thread elements (FRTEs) belonging to a particular file within a given transaction and a given environment. If a transaction accesses several files from within the same environment, there will be one FLAB for each file. If a transaction accesses the same file from more than one environment, there will be one FLAB for each environment.

The FLAB contains pointers to the FCTE for the file, to the owning FRAB, to the chain of FRTEs owned by the FLAB, and to the next FLAB in the chain of FLABs for the unit of work.

The FLAB is used by file control to

- anchor the FRTEs for the file within the unit of work and environment,
- ensure that a file cannot be closed if there are any FRTEs associated with it, or if there have been recoverable updates made by units of work which have not yet reached syncpoint phase 2,

- ensure that the corresponding file entry cannot be reallocated to a different data set, even if the file is closed and disabled, when there is uncommitted recoverable work associated with the file,
- hold READ SET storage control information across intermediate syncpoints,
- ensure that units of work which have updated the file reach syncpoint before a copy or BWO copy for a file opened in RLS mode is allowed to proceed,
- record the reason for a failure during syncpoint, and keep track of the fact that the file has uncommitted updates within a unit of work as a result of the failure.

The file lasting access block is built by DFHFCFR as part of processing of the first file control request for a particular file within a given transaction and environment. FLABs for recoverable files are also rebuilt by DFHFCIR at warm and emergency restart.

The storage for the FLAB is obtained from a FLAB storage subpool above the 16MB line.

The FLAB is deleted after all the FRTEs have been processed during syncpoint terminate processing, providing that there have been no syncpoint failures for the file within the unit of work. The FLAB storage is not returned to the FLAB storage subpool, but is instead added to a chain of free FLABs, anchored from file control static storage. Subsequent requests to build a FLAB are, if possible, satisfied by a quick cell mechanism from this chain.

If a unit of work is shunted as a result of a syncpoint failure, the FLABs for any files which suffered the syncpoint failure are also shunted.

The chain of FLABs for a unit of work is anchored from field FRAB_FLAB_CHAIN_ADDRESS in the FLAB.

The layout of the FLAB is defined by the DFHFLAB structure and the DFHFLAB DSECT.

File control locks locator blocks (FLLBs)

The file control locks locator block records the fact that a unit of work held locks against a file which were protecting uncommitted changes to the file, and that it is now uncertain whether the locks are valid. This can occur, for example, if the data set against which the locks were held is now in the lost locks state, or if a non-RLS open for update has taken place despite the presence of retained locks and has overridden the locks (in this case the locks are intact, but the data may not be). It is used by file control to keep track of outstanding recovery work, because whilst the data set still has FLLBs associated with it, special processing rules apply (the actual rules vary with the type of lock condition that has occurred).

FLLBs are created by DFHFCRR (for the lost locks condition, or for an OFFSITE=YES CICS restart), or by DFHFCRO (after a file open which has returned the ‘non-RLS override’ reason code).

FLLBs are chained from both the associated DSNB and the associated FRAB. There is one FLLB per file that held locks per unit of work. Since the FLLB records information about a data set and a unit of work, it contains the DSNB address and the local unit of work ID. It also contains an indicator of the type of lock failure condition that it represents.

FLLBs are getmained from an FLLB subpool above the 16MB line.

File control locks locator blocks are freemained by DFHFCRC at commit time when there are no longer any retained FLABs for the file.

The layout of the FLLB is defined by the DFHFLLB structure and the DFHFLLB DSECT.

File request anchor block (FRAB)

The file request anchor block serves as an anchor for the set of file lasting access blocks (FLABs) belonging to a particular transaction. The file request thread elements (FRTEs) are chained from the FLABs. The FRAB identifies the transaction to which a given file control request belongs.

The FRAB contains pointers to: the next FRAB in the chain from the FC static, the chain of FLABs for this transaction, the chain of FLLBs for the transaction, and any VSWA that has suffered exclusive control conflict for the transaction. The FRAB also contains some indicators related to recovery, such as whether or not the transaction holds RLS locks, whether the unit of work has been through phase 2 of syncpoint, and whether the unit of work has ever been shunted. There is also some information related to RLS access, including the local unit of work id, a timeout value to be specified on RLS requests, and some problem determination information returned by VSAM RLS when deadlocks occur.

The FRAB is built by DFHFCFR as part of processing of the first File Control request in a transaction. The storage for the FRAB is obtained from a FRAB storage subpool above the 16MB line. The address of the FRAB is then used as the Recovery Manager token associated with the client name ‘FC’. FRABs are rebuilt by DFHFCIR at warm or emergency restart, for units of work which had not completed when CICS terminated. A FRAB is also built if a failure occurs during phase 2 of an intermediate syncpoint. The original FRAB for the transaction is shunted along with the failed parts of the unit of work, and the newly built FRAB is passed on to the next unit of work in the transaction.

If a unit of work is shunted, the FRAB is shunted with it, unless there was no recoverable file control work in the unit of work.

The FRAB is deleted after all the FLABs have been processed during syncpoint at transaction termination. At the same time, the Recovery Manager token is set to zero. At this point, the FRAB storage is not returned to the FRAB storage subpool, but is instead added to a chain of free FRABs, anchored from file control static storage. Subsequent requests to build a FRAB are, if possible, satisfied by a quick cell mechanism from this chain.

Issuing an INQUIRE_WORK_TOKEN call to the recovery manager with client name ‘FC’ returns the address of the file request anchor block for a transaction. There is a chain of all the FRABs in a CICS system, anchored from field FC_FRAB_CHAIN in file control static storage.

The layout of the FRAB is defined by the DFHFRAB structure and the DFHFRAB DSECT.

File request thread elements (FRTEs)

FRTEs are used by file control to:

- Represent active file control requests
- Link related requests together as a file thread, for example, the request sequence STARTBR, READNEXT, ..., ENDBR, or READ UPDATE, REWRITE
- Anchor SET storage used for READ SET UPDATE requests and browse requests with the set option, the lifetime of which is that of the request thread.

FRTEs are created by the main file control module, DFHFCFR, and are freed ***either*** by DFHFCFR at the end of a request or thread of requests ***or*** by the file control recovery control program, DFHFCRC, at syncpoint if this occurs before a thread of requests has completed.

FRTEs for a particular file within a particular task and environment are chained together, and anchored from the FLAB for that file, task and environment.

Storage for FRTEs is acquired from above the 16MB line.

The layout of FRTEs is defined by the DFHFRTE structure and by the DFHFRTE DSECT.

Keypoint list element (KPLE)

The keypoint list forms part of file control's implementation of backup while open (BWO) copy for data sets accessed in non-RLS mode. One KPLE exists for each keypoint and records the start and end times at which tie up records are written.

The KPLE chain is anchored from FC_KPLE_CHAIN in file control static storage.

The keypoint list elements are created, processed and deleted (when they become redundant) by DFHFCRC following RMKP take keypoint calls from the recovery manager. These calls are made whenever a CICS keypoint is taken. KPLEs are getmained from above the 16MB line.

The layout of the KPLE is defined by the KPLE structure.

Shared resources control (SHRCTL) block

The SHRCTL block represents the CICS region's requirements of, and the use made of, a local shared resources pool (LSRPOOL). It is used by DFHFCL when calling VSAM to build an LSRPOOL. It is also used by DFHFCL and statistics programs to hold and update file control statistics. It lasts for the lifetime of a CICS run, and is addressable through a pointer in file control static storage. There are eight pointers collectively named the SHRCTL vector.

A SHRCTL block holds information such as how many virtual and hyperspace buffers of a particular size are needed, how many strings are needed, the maximum key length allowed. CICS passes this information to VSAM when the pool is built. It also holds statistics about the pool which are sent to the statistics domain when requested or when the pool is deleted.

Each SHRCTL block represents one LSRPOOL, and there are eight SHRCTL blocks. The layout of each SHRCTL block is defined by the DFHFCTLS structure and by the DFHFCTSR DSECT, and they reside above the 16MB line.

On a CICS cold start, DFHFCRP performs the following:
- Unconditionally builds eight SHRCTL blocks above the 16MB line (from a SHRCTL block subpool)
- Fills in default settings in the block, or inserts user-specified information
- Catalogs each SHRCTL block in the CICS global catalog (GCD).

On a CICS warm or emergency start:
- DFHFCRP restores all eight SHRCTL blocks from the global catalog.

The contents of a SHRCTL block are decided in one of three ways:
- User defines the contents in the FCT by means of the DFHFCT TYPE=SHRCTL,LSRPOOL=n macro call. This assembled information is used by DFHFCRP on a COLD start only (as per FCT entries).
- User defines the contents online through a CEDA DEFINE LSRPOOL command.
- If neither of the above two methods is used, DFHFCL calculates the contents before calling VSAM to build the LSRPOOL.

VSAM work area (VSWA)

The VSWA represents a VSAM request to CICS. Embedded in the VSWA is the request parameter list (RPL) which is passed to VSAM to perform the request. In addition to the RPL, the VSWA contains other CICS information related to the request.

The VSWA is used by DFHFCVS and DFHFCRS when processing VSAM files.

A VSWA survives as long as the RPL needs to survive to complete the VSAM request; for example, it survives from READ UPDATE to the REWRITE request.

File control

The address of the VSWA is held in the file request thread element (FRTE) in the FRT_WORK_AREA_ADDRESS field.

Storage for the VSWA is acquired from above the 16MB line.

The layout of the VSWA is defined by the DFHVSWAS structure and by the DFHVSWA DSECT.

Modules

This Section describes the following modules. Unless otherwise stated, addressing mode and residency mode are AMODE 31 and RMODE ANY respectively.

Module	Function	See page
DFHFCVR	File control VSAM interface program	214
DFHFCVS	File control VSAM request processor	215

There are also a number of modules which make up the coupling facility data tables server. These all have names of the form DFHCFxx.

Figure 50 shows the main file control modules and their interfaces.

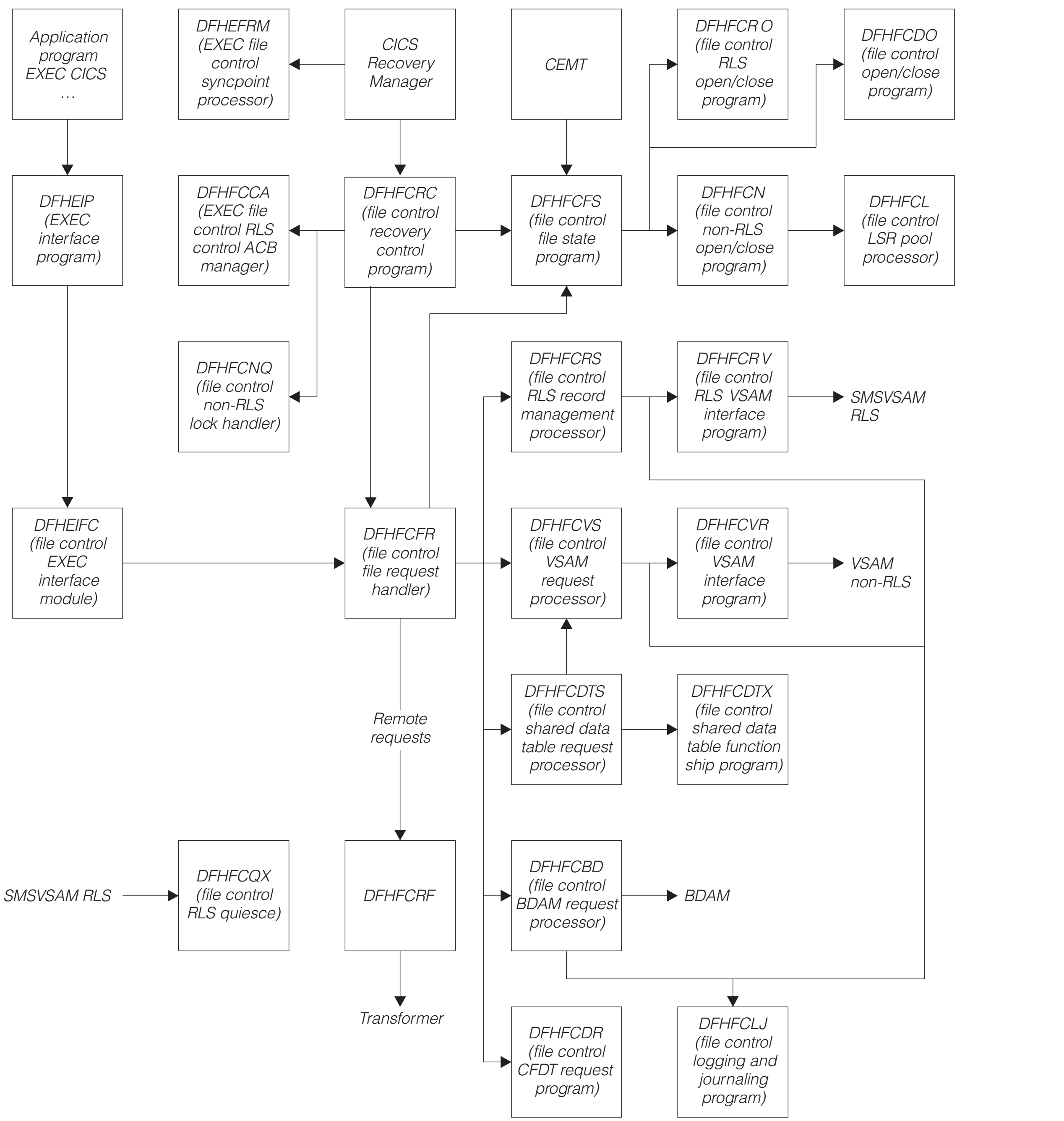


Figure 50. Main file control modules and their interfaces

DFHEIFC (file control EXEC interface module)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address
DFHEIFC. Stored in the CSA in a field named CSAEIFC.

Purpose
DFHEIFC is DFHEIP's file control interface. It routes requests to the file control file request handler, DFHFCFR.

Called by
DFHEIP exclusively.

Inputs
The EIEI parameter list, as defined by the DFHEIEIA DSECT.

Outputs
Updated EIEI parameter list, with completed EIB.

Operation
- Call user exit XFCREQ.
- Call file control request handler DFHFCFR.
- Call user exit XFCREQC.

How loaded
At CICS startup, as part of the building of the CICS nucleus. The nucleus is built by DFHSIB1, which uses its nucleus build list to determine the content and characteristics of the CICS nucleus.

DFHFCAT (file control catalog manager)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address
DFHFCAT. The entry point address is held in FC static storage in a field named FC_FCAT_ADDRESS, which is set by DFHFCRP when it loads DFHFCAT.

Purpose
The file control catalog manager is part of the file control component. This program processes inquire and update requests on the state of the backup while open (BWO) attributes in the ICF catalog for VSAM data sets and inquire on the quiesce state in the ICF catalog. The DFSMS Callable Services interface is used for these operations.

Called by

DFHFCDN	Get the base data set name for a DSNB that has not yet been validated, update the recovery point, or to set the BWO attributes to a 'forward recovered' state
DFHFCN	Inquire on the current state of, and to update, BWO attributes during file open processing; and to reset these attributes during file close processing.
DFHFCQI	Inquire on the quiesce state of a data set.

Inputs
The FCAT parameter list, as defined by the DFHFCATA DSECT, is created as part of the subroutine call.

The input parameters are:

Data set name
Recovery point

Outputs

Returned in the FCAT parameter list:
Quiesce state
Base data set name
State (fuzzy, sharp)
Response
Reason

Operation

DFHFCAT provides the following functions:

INQ_BASEDSNAME
: Gets the base data set name for a specified data set name from the ICF catalog. This function is used when there is not a validated DSN block for the data set.

INQ_CATALOG_QUIESCESTATE
: If the level of DFSMS is 1.3 or higher, issues an IGWARLS call to determine the quiesce state of the data set (quiesced or unquiesced).

INQ_DATASET_STATE
: Determines the current state of a VSAM data set's BWO attributes in the ICF catalog. If the BWO attributes indicate that the data set is "back level", that is, a backup copy has been restored but not forward recovered, an exception response is returned; otherwise, a state of 'fuzzy' or 'sharp' is returned, indicating whether or not the data set is defined in the ICF catalog as eligible for BWO.

SET_CATALOG_RECOVERED
: Updates a VSAM data set's BWO attributes in the ICF catalog to a 'forward recovered' state to indicate that the data set has been forward recovered.

SET_CATALOG_RECOV_POINT
: Updates a VSAM data set's BWO attributes in the ICF catalog with the new recovery point.

SET_BWO_BITS_DISABLED
: Updates a VSAM data set's BWO attributes in the ICF catalog to show that the data set is no longer eligible for BWO support, and updates the recovery point.

SET_BWO_BITS_ENABLED
: Updates a VSAM data set's BWO attributes in the ICF catalog to show that the data set is eligible for BWO support, and updates the recovery point.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCBD (file control BDAM request processor)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCBD. The entry point address is held in FC static storage in a field named FC_BDAM_ENTRY_ADDRESS.

Addressing mode

AMODE 31.

Residency mode

RMODE 24.

Purpose
The BDAM request processor is part of the file control component. It processes access requests to BDAM files.

Called by
DFHFCFR, after having determined that the request is for a BDAM file.

Inputs
The FCFR parameter list, as defined by the DFHFCFRA DSECT. Also, the file control environment, including FC static storage and the FCT.

Outputs
Updated FCFR parameter list.

Operation
Acquires and releases FIOA storage as necessary. Implements BDAM exclusive control requests. Performs record-length and key-length checking. Calls BDAM to perform the I/O request.

Acquires storage, in the correct key subpool, for requests that specify SET.

How loaded
By DFHFCFS, by means of a loader domain call. DFHFCBD is not loaded unless DFHFCFS is called to open a BDAM file and, in doing so, it discovers that DFHFCBD is not yet in storage.

DFHFCCA (file control RLS control ACB manager)
DFHFCCA is the file control RLS control ACB manager. The RLS control ACB is a special ACB required when a commit protocol application such as CICS uses VSAM RLS. FCCA processes requests to register and unregister the control ACB, and all other file control requests to SMSVSAM that have to be made via the control ACB. These requests are:
- IDAREGP (register)
- IDAUNRP (unregister)
- IDARECOV (clear recovery status)
- IDAINQRC (inquire on recovery)
- IDAQUIES (quiesce)
- IDALKREL (release locks, and retain locks marked for retention)
- IDARETLK (mark locks for retention)

DFHFCCA also includes the code for the RLSWAIT exit used by control ACB requests. Whenever CICS issues such a request, VSAM drives the RLSWAIT exit as soon as it is about to transfer control to the SMSVSAM address space. CICS is then able to drive the dispatcher and schedule other CICS tasks whilst the SMSVSAM address space is busy processing the request.

DFHFCDL (file control CFDT load program)
DFHFCDL is attached by DFHFCDO to load a load-capable coupling facility data tavle with records from a source data set.

DFHFCDN (file control DSN block manager)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCDN. The entry point address is held in FC static storage in a field named FC_FCDN_ADDRESS, which is set by DFHFCRP when it loads DFHFCDN.

Purpose

The DSNAME block manager is part of the file control component. This program is called to perform various operations on data set name blocks. These operations include connecting and disconnecting DSN blocks and FCT entries, setting their attributes, and deleting them when no longer required. The program also allows the caller to inspect a particular DSN block or browse a set of blocks. It can also be called to update the backup while open (BWO) attributes in the ICF catalog for VSAM data sets, and to set the quiesce state to normal in all DSN blocks. Finally it can be called to catalog the information in a DSN block to the CICS global catalog.

Called by

DFHAMFC Connect a DSN block to a newly created FCT entry

DFHAMPFI Connect the DSN block for the CSD to the associated FCT entry

DFHEIQDN Connect, disconnect, delete, set attributes, browse, and inquire against DSN blocks in response to external requests; and to update the BWO attributes in the ICF catalog for a VSAM data set to a 'forward recovered' state

DFHEIQDS Connect or disconnect DSN blocks and FCT entries in response to external requests

DFHFCLF Set the availability attribute to unavailable after a forward recovery log stream failure

DFHFCMT Disconnect the DSN block when deleting an FCT entry

DFHFCN Connect or disconnect and to catalog a DSN block

DFHFCRC Update the recovery point in the ICF catalog for all VSAM data sets that are open for update in non-RLS mode and defined as eligible for BWO support at keypoint time

DFHFCRD To reset all quiesce states to normal after an SMSVSAM server failure

DFHFCRO Connect or disconnect and to catalog a DSN block

DFHFCRP Connect or reconnect DSN blocks during file control initialization or restart.

Inputs

The FCDN parameter list, as defined by the DFHFCDNA DSECT, is created as part of the subroutine call.

The input parameters include:

- Request identifier
- Address of FCTE or FCTE token
- Data set name
- Browse token
- Availability status
- Type of pointer
- Recovery point

Outputs

Output parameters, as part of the FCDN parameter list. Apart from the response, all these are returned on the inquire or browse requests. The parameters include:

Access method
Base data set name
Availability status
DSNB type
File count
DSNB valid status
Lost locks status
Forward-recovery log stream name
Forward-recovery log ID

Recovery status
Response
Reason

Operation

- Connect:

  The inputs are a data set name and an FCTE pointer or an FCTE token, with an indication of whether the entity to be connected is a base or an object.

  If the FCT entry is already connected, the connection is broken before connecting it to a DSN block representing the new object. The DSN block that is connected can exist already, or DFHFCDN creates a new block before connecting it.

  The request is rejected if it requires an existing connection to be broken, and there are uncommitted updates to the file; that is, there are retained locks.
- Disconnect:

  The connection between the FCT entry and the DSN block is broken. The DSN block remains even if there are no other FCT entries connected to it. The request is rejected if there are uncommitted updates to the file: that is, there are retained locks.
- Delete:

  Checks are made to ensure that the DSN block is allowed to be deleted. If the deletion can proceed, the table manager is called to delete the DSN from the DSN index, and the storage domain is called to free the storage.
- Inquire:

  The attributes stored in the DSN block are returned to the caller in the FCDN parameter list.
- Set:

  The availability status is set in the DSN block. The catalog domain is called to catalog the change.
- Start browse, get next, end browse:

  The DSN blocks are browsed in order. For each, the attributes are returned to the caller.
- Catalog:

  The information in a DSN block is cataloged to the CICS global catalog.
- SET_CATALOG_RECOVERED:

  This function is used by DFHEIQDN. DFHFCDN in turn issues a SET_CATALOG_RECOVERED call to DFHFCAT to update the BWO attributes in the ICF catalog for a given VSAM data set to a ‘forward recovered’ state.
- UPDATE_RECOVERY_POINTS:

  This function is used by DFHFCRC. DFHFCDN in turn issues a SET_CATALOG_RECOV_POINT call to DFHFCAT to update the recovery point in the BWO attributes in the ICF catalog for every data set that is open for update in non-RLS mode and defined as eligible for BWO support.

  The recovery point is the time from which a forward-recovery utility should start applying log records. It is always before the time the last backup was taken. For further information about recovery points and backup while open in general, see the *CICS Recovery and Restart Guide*.
- RESET_ALL_QUIESCE_STATUS:

  This function is used by DFHFCRD. The DSNB table is scanned, and the quiesce status is reset to normal in each DSNB.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCDO (file control CFDT open/close program)

When called using the FCFS parameter list, DFHFCDO performs an equivalent function for coupling facility data table opens and closes as is performed by DFHFCN for non-RLS VSAM files.

When called using the FCDS parameter list, DFHFCDO performs statistics collection for coupling facility data tables, and disconnects from CFDT pools at shutdown.

DFHFCDR (file control CFDT request processor)

DFHFCDR performs an equivalent function for coupling facility data tables as is performed by DFHFCVS for non-RLS VSAM files, and uses the same interface.

DFHFCDTS (file control shared data table request program)

DFHFCDTS performs an equivalent function for CICS-maintained and user-maintained data tables as is performed by DFHFCVS for non-RLS VSAM files and uses the same interface.

DFHFCDTX (file control shared data table function ship program)

DFHFCDTX receives file requests from DFHFCDTS in FCFRR format, converts them into command level interface form and then calls ISP to function ship the request.

The response returned by ISP in the EIB is translated back into an FCFRR response and reason code.

DFHFCDU (file control CFDT UOW calls program)

DFHFCDU encapsulates the processing required to call the coupling facility data tables server for unit of work related operations, such as commit, backout, inquire. It is called via the FCDU parameter list by DFHFCDW and DFHFCDY.

DFHFCDW (file control CFDT RMC program)

DFHFCDW provides a recovery manager connector (RMC) between file control and the coupling facility data tables server, to support 2-phase commit and recovery for recoverable coupling facility data tables. It is called by the CICS Recovery Manager using the RMLK parameter list.

DFHFCDY (file control CFDT resynchronization program)

DFHFCDY performs resynchronization of coupling facility data table pools and links. It is called using the FCDY parameter list by DFHFCDO, DFHFCDR and DFHFCDU.

DFHFCES (file control ENF servicer)

DFHFCES is the file control ENF servicer. It is used to prompt dynamic restart of RLS file control when the SMSVSAM Server becomes available again after an earlier failure. DFHFCES is invoked whenever the MVS Event Notification Facility notifies CICS (via the CICS domain manager ENF support) that SMSVSAM is available.

DFHFCES establishes a transaction environment, and calls DFHFCRR to dynamically restart RLS.

DFHFCFL (file control FRAB and FLAB processor)

DFHFCFL is the File Control FRAB/FLAB processor. It contains a number of functions to process FLAB control blocks belonging to a particular base data set. It processes the functions of the FCFL interface.

The DSNB of the data set is not locked during the processing of the commands. As a FLAB exists, and hence an FCTE, the DSNB cannot be deleted, therefore there is no need to lock the DSNB.

DFHFCFR (file control file request handler)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCFR. Stored in the CSA in a field named CSAFCEP.

Purpose

The central module in the file control component.

Processes file control requests issued by DFHEIFC (requests from application programs), or from other CICS modules (internal CICS file control requests).

Receives and routes file control access-method dependent requests to one of the following:

- DFHFCRS for VSAM RLS files
- DFHFCVS for VSAM non-RLS files
- DFHFCBD for BDAM files
- DFHFCDR for coupling facility data tables
- DFHFCTS for user-maintained data tables
- DFHFCDTS for non-update requests to CICS maintained data table
- DFHFCVS for update requests to CICS-maintained data tables
- DFHFCRF for requests to remote files

Implements TEST_FILE_USER requests.

Routes RESTART_FILE_CONTROL requests to DFHFCVS and DFHFCRS during the file control initialization.

Frees buffers at the request of DFHAPSM when 'short on storage' has been detected.

Performs a CLEAR_ENVIRONMENT when requested by DFHERM, DFHAPLI or DFHUEH. This cleans up file control storage at the completion of a task-related user exit, a user-replaceable program, or a global user exit:

- The FLAB and FRTE chain are scanned to find all FRTEs for the specified environment.
- An ENDBR request is issued to terminate any active browse operation.
- An UNLOCK request is issued for any active READ UPDATE or WRITE MASSINSERT.

Called by

DFHAPLI	AP language interface program
DFHAPSM	AP domain storage notify gate
DFHDMPCA	CSD manager adapter
DFHDTLX	Shared data tables load program
DFHEIFC	File control EXEC interface module
DFHERM	Resource manager interface (RMI) module
DFHFCDL	Coupling facility data tables load program
DFHFCDTS	File control shared data table request processor
DFHFCFR	File control file request handler (a recursive call)
DFHFCRC	File control recovery control program
DFHFCRP	File control restart program
DFHUEH	AP user exit handler.

Inputs

The FCFR parameter list, as defined by the DFHFCFRA DSECT. Also the file control environment, including FC static storage and the FCT.

Outputs

Updated FCFR parameter list.

Operation

Selects on the request type, and passes control to the routine specific to that request.

Performs monitoring.

Obtains a FLAB and FRTE to represent this request, or scans the FLAB and FRTE chains to associate this request with a previous FRTE if required. Some checking for error situations is performed during the scan.

Performs file state checking to determine whether or not a (VSAM or BDAM) request to a file is able to proceed. If file is enabled but closed and is not a request to a remote file, opens it before carrying out the request.

Checks for “privileged” requests.

If the request is not remote, checks the “service request” attributes for the file to determine whether the request can proceed.

Checks the file’s access method (VSAM or BDAM as defined in the FCT). If BDAM, calls DFHFCBD to process the request. If VSAM and non-RLS, calls DFHFCVS to process the request. If VSAM and RLS, calls DFHFCRS to process the request. If a data table, calls DFHFCDTS for read requests against a CICS-maintained data table or any request against a user-maintained table, and calls DFHFCVS otherwise (that is, for update and browse requests against a CICS-maintained data table). If the file is remote, calls DFHFCRF to process the request.

On return, performs cleanup if required.

How loaded

By DFHSIB1 as part of the CICS nucleus.

DFHFCFS (file control file state program)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCFS. The entry point address is held in FC static storage in a field named FC_FCFS_ADDRESS, which is set by DFHFCRP when it loads DFHFCFS.

Purpose

The file control file state program is part of the file control component.

The program processes requests to enable, disable, open, and close files. Such requests can originate from explicit requests (either CEMT or EXEC CICS SET), from implicit requests (such as implicit open), or from requests made from CICS internal processing.

Close and disable requests are processed in different ways, depending on whether the request has been issued with the WAIT or the NOWAIT option. A request with the WAIT option is treated as a synchronous request, that is, control returns to the requesting program only after all users of the file have completed their use.

A request with the NOWAIT option is treated as an asynchronous request. In this case, the file is marked with the intended state and control is returned immediately.

Called by

DFHAMFC	Enable a newly installed file

DFHDMPCA	Change the state of the CSD
DFHDMRM	Close CSD after an error
DFHDTLX	Close the data set associated with a shared data table
DFHEIQDS	Implement CEMT and EXEC CICS requests
DFHFCDL	Close the data set associated with a coupling facility data table
DFHFCDTS	Close shared data table if remote connection disabled or invalidated
DFHFCFR	Implicit open
DFHFCQU	Close files for quiesce, cancel close for unquiesce, enable files
DFHFCRC	Open files which need backout, and close files at syncpoint
DFHFCRD	Immediate close of RLS files
DFHFCRV	Close files for pending immediate close requests
DFHFCSD	Close files on a normal CICS shutdown
DFHFCU	Open all files with FILSTAT=OPEN coded
DFHFCVS	Open the base, and during empty file or I/O error processing.

Inputs

The FCFS parameter list, as defined by the DFHFCFSA DSECT, is created as part of the subroutine call.

The input parameters are:

Request identifier (open, close, enable, disable, cancel close)
FCTE address
FCTE token
Open options (open base, open for backout)
Close qualifier (close pending, shutdown, immediate close,
quiesce, and so on)
Action (wait, do not wait, force)

Outputs

Returned in the FCFS parameter list:

DFHFCN return code
Register 15 return code
VSAM return code

Operation

Before any processing to change the state of a file is carried out, its FCT entry is locked by means of a DFHKC ENQ call. At the conclusion of file state change processing, the FCT entry is unlocked before returning to the caller.

- Enable file.

  DFHFCFS marks the FCT entry as ‘enabled’, and catalogs the change.
- Disable file.

  If the WAIT option is specified, DFHFCFS tests whether the transaction issuing the request is a current user of the file. If it is, DFHFCFS returns an exception response.

  DFHFCFS next marks the FCT entry entry as ‘disabled’ and catalogs the change. If the disable request stems from a close request (see later), DFHFCFS also sets the implicit indicator, thereby marking the state as ‘unenabled’. However, if this close request originated from DFHFCSD as part of CICS shutdown processing, DFHFCFS does ***not*** mark the state as ‘unenabled’.

  Finally, if the WAIT option is specified, the FCT entry is unlocked before waiting for the ‘disabled’ ECB in the FCT entry to be posted by the transaction that reduces the use count to zero.
- Open file.

  If the file is unenabled (due to a previous close), DFHFCFS enables it and catalogs the new state, unless the open option is open for backout.

If the file refers to a BDAM data set, DFHFCFS tests whether DFHFCBD is already loaded; if not, it calls loader domain to do so.

If the file is a data table, DFHFCFS loads and initializes data table services, if this has not been done already on a previous open request.

DFHFCFS next calls DFHFCN (for non-RLS) or DFHFCRO (for RLS) to perform the physical open. After the file has been successfully opened, its FCT entry is marked accordingly.

For a data table, DFHFCFS issues OPEN and LOAD requests to data table services.

- Close file.

  If there is no close qualifier, the file is first implicitly disabled (as described above), taking into account the WAIT or NOWAIT option. The new state is cataloged.

  If the file use count is zero, DFHFCFS calls DFHFCN or DFHFCRO to perform the physical close. After the file has been successfully closed, its FCT entry is marked accordingly.

  An immediate close is issued if the SMSVSAM RLS server fails. The close must wait until there are no requests active in the RLS record management processor. The enablement state of the file is not changed. A close with close qualifier of quiesce is issued to process an RLS quiesce request. The file is unenabled, and the state catalogued.

  For a data table, DFHFCFS issues a CLOSE request to data table services, except in the case of a special type of CLOSE request issued by DFHFCVS for a user-maintained data table, when loading is complete and the source data set is to be closed, but not the table itself.

  For a remote data table, DFHFCFS issues a DISCONNECT request to data table services.

  If the file use count is nonzero, DFHFCFS sets the ‘close requested’ indicator in the FCT and returns to the caller. Any subsequent transaction that reduces the use count to zero tests the ‘close requested’ indicator and, if set, performs the actual close.

  When called by DFHFCSD during CICS shutdown, DFHFCFS ensures that files are closed, marks the file as ‘closed unenabled’ in the FCT, but does ***not*** record this change in the global catalog. This allows implicit file opens on a subsequent restart.
- Cancel close.

  An in-progress close is cancelled if a data set is unquiesced. The close_in_progress flag is reset, any tasks waiting for the file to close are resumed, and the file is re-enabled.

How loaded
By DFHFCRP as part of file control initialization.

DFHFCIN1 (file control initialization program 1)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address
DFHFCIN1. Stored in the CSA in a field named CSAFCXAD.

Purpose
The file control initialization program is part of the file control component. This program initializes file control and starts the file control restart task. It also waits for the restart task to complete, and returns the status of the completion to the caller.

Called by
DFHSII1, as part of CICS initialization.

Inputs
The FCIN parameter list, as defined by the DFHFCINA DSECT.

Outputs

Updated FCIN parameter list.

Operation

Initialize:

- Calls storage manager domain to add a subpool for file control static storage.
- Calls storage manager domain to create the storage for file control static storage.
- Initializes file control static storage.
- Attaches the file control restart task by means of a DFHKC request, with entry point address DFHFCIN2.

WAITINIT:

- Issues a dispatcher domain call to wait on the CICS ECB which indicates that the file control restart task has finished (FC_RECOV_ALLOWED_ACB) in file control static storage.
- On completion of the wait, tests the response and returns to DFHSII1.

How loaded

Link-edited with DFHFCIN2 to form the DFHFCIN module, which is loaded by DFHSIB1 as part of the CICS nucleus.

DFHFCIN2 (file control initialization program 2)

Call mechanism

Attached by DFHFCIN1 as a separate CICS task. Given control by means of the DFHKC TYPE=ATTACH mechanism.

Entry address

DFHFCIN2. Because DFHFCIN2 is link-edited with DFHFCIN1, the entry address is known to DFHFCIN1 at the time the DFHKC TYPE=ATTACH is issued.

Purpose

The file control initialization program is part of the file control component. This program loads and calls the file control restart program (DFHFCRP), to perform file control restart as a separate task.

Called by

CICS task control, after being attached by DFHFCIN1.

Inputs

None.

Outputs

The initialized file control component. Addresses and indicators completed in file control static storage.

Operation

Calls loader domain to acquire (that is, to load) the DFHFCRP program. Stores the entry point address of the loaded module (which is also the load point) in DFHFCIN2's automatic storage in a field named FCRP_ENTRY_ADDRESS.

If the ACQUIRE request failed, calls loader domain to define program and then retries the ACQUIRE request.

Calls DFHFCRP by means of a subroutine call via the kernel.

On successful completion, calls loader domain to release DFHFCRP. On both successful and unsuccessful completion, posts the ECBs FC_NON_RECOV_ALLOWED_ECB and FC_RECOV_ALLOWED_ECB. The success or otherwise of File Control restart is indicated by the flag FCSCMPLT in file control static storage.

On unsuccessful completion, posts the Restart Task ECB complete and returns.

How loaded

By DFHSIB1 as part of the CICS nucleus.

DFHFCIR (file control initialize recovery)

DFHFCIR is the File Control Initialize Recovery Module. It initializes the File Control environment in which recovery after a CICS failure is carried out.

DFHFCIR handles the delivery of recovery data by the CICS Recovery Manager during its scan of the system log at warm or emergency restart, and rebuilds the file control structures that represent units of work that were in-flight or shunted when CICS terminated.

During its log scan, Recovery Manager calls File Control's recovery gate, which invokes the module DFHFCRC. DFHFCRC passes the calls through to DFHFCIR via a kernel subroutine call. The calls are the RMDE functions START_DELIVERY, DELIVER_RECOVERY, DELIVER_FORGET and END_DELIVERY.

DFHFCL (file control shared resources pool processor)

Call mechanism

BALR, obtaining LIFO storage on entry.

Entry address

DFHFCLNA. DFHFCL is, together with DFHFCN and DFHFCM, link-edited with DFHFCFS. All calls to DFHFCL are made from DFHFCN; the entry point address is known to DFHFCN from the link edit.

Purpose

The shared resources pool processor is part of the file control component.

This program is called at file open time to create a specific local shared resources pool if it does not exist. It is also called to delete a specific pool when the last file to use the pool is being closed.

The size and characteristics of the pool being built are obtained either from information in the SHRCTL definition in the FCT or, if that information has not been provided, from the best information available to DFHFCL at the time of the open.

Called by

DFHFCL is called exclusively by DFHFCN.

Inputs

The FCLPARAM parameter list, created in DFHFCN's automatic storage and addressed by register 1 on the call.

The input parameters are:

Request identifier (build, delete)
LSR pool number

Outputs

Returned in the FCLPARAM parameter list:

DFHFCL return code
BLDVRP/DLVRP return code
VSAM return code

Operation

If the request is for LSR pool creation, DFHFCL first checks whether the SHRCTL block includes specifications for the number of strings, maximum key length, and the number of virtual and hyperspace buffers of each of the eleven sizes in the pool. If these values are known, DFHFCL sets up the BLDVRP parameter list and creates the pool by issuing the BLDVRP macro.

If some or all of the pool characteristics are not specified in the SHRCTL definition, DFHFCL calculates the pool requirements from the information in the FCT and the VSAM catalog.

Each FCT entry is inspected to find whether it is to be included in the pool being built. If so, its DSNAME is determined and this is used to obtain data set characteristics from the VSAM catalog. The information required for the BLDVRP macro is accumulated in the SHRCTL block and the pool is built from these values.

If the request is for LSR pool deletion, DFHFCL first obtains the VSAM statistics for the pool and saves them in the SHRCTL block. These statistics are unobtainable after the pool has been deleted.

DFHFCL next deletes the specified pool by issuing a DLVRP macro.

Finally, DFHFCL sends pool statistics to the statistics domain as unsolicited data.

How loaded

As a constituent part of DFHFCFS, which is loaded by DFHFCRP as part of file control initialization.

DFHFCLF (file control log failures handler)

DFHFCLF provides control of long term logger failures for File Control. It is called in the event of a failure of a general log stream, which will be either the forward recovery log for a data set or the autojournal for a file.

The CICS Log Manager invokes DFHFCLF when an MVS log stream being used for forward recovery or file autojournalling suffers a long term failure. The call is made using the LGGL ERROR function.

When file control opens a forward recovery log stream or an autojournal, it will register this call back gate to the Log Manager by specifying FCLF as the file control error gate.

When called, DFHFCLF takes action to ensure that the log stream failure causes minimum damage. For a forward recovery log failure it closes all files open against the data set using that forward recovery log (across the sysplex for a data set accessed in RLS mode) and issues a message advising that a new backup copy should be taken. For an autojournal it closes the file using that autojournal and issues a warning message.

DFHFCLJ (file control logging and journaling program

DFHFCLJ is the file control logging and journaling program. It is called to perform logging for transaction backout and forward recovery, to write to journals for autojournal requests and to write to the log of logs.

Records are written to the system log using the RMRE APPEND function, and optionally forced using the RMRE FORCE function. Records are written to forward recovery logs and autojournals using the LGGL WRITE function, and to the log of logs using the LGGL WRITE_JNL function.

DFHFCMT (file control table manager)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address
DFHFCMT. The entry point address is held in FC static storage in a field named FC_FCMT_ADDRESS, which is set by DFHFCRP when it loads DFHFCMT.

Purpose
The file control table manager is part of the file control component. This program is called to add, delete, and set FCT entries, and to return attributes of an FCT entry (inquire).

Called by
`DFHAMFC`	Inquire on, add, or delete a newly created FCT entry to the system
`DFHAMPFI`	Add the entry in the FCT for the CSD to the system
`DFHDMPCA`	Inquire on and set the attributes of the FCT entry for the CSD
`DFHEDFX`	Inquire on the attributes of an FCT entry
`DFHEIQDS`	Inquire on or set the attributes of FCT entries, or delete an FCT entry.

Inputs
The FCMT parameter list, as defined by the DFHFCMTA assembler DSECT, is created as part of the subroutine call.

The input parameters are:

```
Common parameters:
    File name
    String number
    Journal ID
    Recovery characteristics
    Journaling characteristics
    Enablement status
    Open time
    Data set disposition
    Service request attributes
    Record format
    Number of data buffers
    Number of index buffers
    Whether to catalog the FCT entry

VSAM-specific parameters:
    VSAM password
    Empty status
    Data set name sharing
    LSR pool ID
    Base name
    Forward recovery log ID
    BWO eligibility
    RLS access mode
    Read integrity

BDAM-specific parameters:
    Exclusive control
```

Outputs

Output parameters, as part of the FCMT parameter list. Apart from the response, all these are returned on the inquire or browse requests. The output parameters are:

Common parameters:
- File type
- String number
- Record size
- Key length
- Key position
- Recovery characteristics
- Journaling characteristics
- Enablement status
- Open status
- Open time
- Data set type
- Data set disposition
- Data set name
- Base data set name
- Service request attributes
- Record format
- Block format
- Access method
- Remote name
- Remote system

VSAM-specific parameters:
- VSAM password
- Empty status
- Object type
- Data set name sharing
- Number of data buffers
- Number of index buffers
- Number of active strings
- LSR pool ID
- Whether using shared resources
- Forward-recovery log ID
- RLS access mode
- Read integrity

BDAM-specific parameters:
- Block size
- Block key length
- Relative address form
- Exclusive control
- Response
- Reason

Data Table specific parameters:
- Table type
- Table size

Operation

- Add:

  Storage for the new FCT entry is obtained out of the VSAM FCT storage subpool (BDAM FCT entries cannot be created).

The new FCT entry is completed by filling in the information from the caller's parameter list.

The name of the new FCT entry is added to the TMP index.

Finally the information in the new entry is written to the CICS global catalog if required.

- Delete:

  The request is rejected if there are uncommitted updates for the file; that is, there are retained locks. DFHTMP is called to locate and quiesce the FCT entry.

  Any DSN block that is connected to the FCT entry is disconnected.

  The FCT entry name is deleted from the TMP index.

  The storage for the FCT entry is freed. In the case of a BDAM FCT entry, its DCB storage is also freed.

  Any catalog entries for the FCT entry are deleted.
- Set:

  DFHTMP is called to locate the FCT entry.

  The request is rejected if there are uncommitted updates for the file; that is, there are retained locks.

  If the FCT entry is not marked 'closed' and 'disabled' (or 'unenabled'), the request is rejected.

  Changes are made to the information in the FCT according to the caller's parameter list.

  Finally the changes are recorded by writing them to the CICS global catalog.
- Inquire:

  DFHTMP is called to locate the FCT entry.

  The attributes are returned in the FCMT parameter list.
- Connect:

  DFHTMP is called to locate the FCT entry.

  The connect count is incremented. The FCT token is returned to the caller.
- Disconnect:

  DFHTMP is called to quiesce the FCT entry.

  A check is made to ensure that the file is closed and disabled (or unenabled). If the check fails, an error is returned to the caller.

  The connect count in the FCT is cleared and a call is again made to DFHTMP to release the quiesce.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCN (file control open/close program)

Call mechanism

BALR, obtaining LIFO storage on entry.

Entry address

DFHFCNNA. DFHFCN is link-edited with DFHFCFS. All calls to DFHFCN are made from DFHFCFS; the entry point address is known to DFHFCFS from the link-edit.

Purpose

The file control open/close program is part of the file control component.

This program performs the physical opening and closing of files by making the corresponding requests to VSAM or BDAM. Associated with these operations are a number of further activities that must be completed before control is returned to DFHFCFS.

These activities include:

- Dynamic allocation of the file

- Empty file checking
- Dynamically setting up ACB fields in advance of the VSAM open
- Copying into file-control control blocks VSAM information about the file which is available after the open
- Inquiring on, and updating, the VSAM data set's backup while open (BWO) attributes in the ICF catalog for a file that is defined in the FCT as eligible for BWO support if the appropriate prerequisite software levels have been installed
- On close, deallocating the file if necessary and clearing the file control information related to the file
- Resetting a VSAM data set's BWO attributes in the ICF catalog during close processing.

Called by

DFHFCFS, exclusively.

Inputs

The FCSPARMS parameter list, created in DFHFCFS's automatic storage and addressed by register 1 on the call.

The input parameters are:

FCTE address
Request identifier

Outputs

Returned in the FCSPARMS parameter list:

DFHFCN return code
Register 15 return code
VSAM return code
Base data set name
Recovery attributes of base

Operation

Execution of the DFHFCN code is serialized. This is done by DFHFCFS issuing a DFHKC ENQ before calling DFHFCN, and a DFHKC DEQ after calling DFHFCN. As a consequence, only a single open or close request to any file can be in progress at any time, and multiple concurrent requests are single-threaded.

The main actions when processing an open request:

1. If the file is being opened for update and any type of autojournalling is specified on the file definition, then the autojournal log stream is opened, via a call to DFHLGGL.
2. The file is tested to determine if it is allocated to the job by means of a JCL statement or is to be allocated dynamically.

   If the file is already allocated, any existing DSN block to which it may be connected is disconnected and a new block with the actual DSNAME is connected. Connecting and disconnecting of DSNAME blocks is always performed by calling DFHFCDN.

   If the file is not already allocated, it is at this point dynamically allocated to the DSNAME in the DSNAME block to which it is connected.

   In the case of a VSAM file, the file's data set name is used to issue appropriate SHOWCAT and LOCATE instructions to determine relevant information from the VSAM catalog about the data set that the file represents. In particular, the following are obtained:

   Base/path indicator
   Base data set name
   Attributes of the data set
   Key length of the base

Relative key position of base key
Maximum record length
Control interval size
Share options
High RBA

3. The data set is checked to determine if it is empty (high RBA is zero) or is to be emptied.

   The ‘load’ mode indicator is set on.

4. DFHFCDN is now called to connect the FCT entry to a DSNAME block for the base cluster (which may be the existing allocation DSNAME block, or may need to be newly created, or may already exist and need only be pointed to from the FCT). The base cluster’s attributes, as obtained from the VSAM catalog, are stored in the base cluster block.

   The file’s recovery characteristics are checked against any that may already have been stored in the base cluster block and, if they have not yet been set up, are saved there. Any conflict with the stored values is handled. In some cases the new value overrides the old one, in others an error is returned.

   During this processing, if this is the first open for update for a file associated with this particular data set:

   a. a call is made to the VSAM callable interface IGWARLS, in order to get any recovery attributes that may be defined in the VSAM catalog. If they are present, then they override any values in the FCT entry.
   b. if forward recovery logging is specified, the forward recovery log stream is opened, using either the log stream name from the VSAM catalog, or a log stream name derived from the id specified in the file definition.

   In the case of an entry sequenced data set or a path to an ESDS, the next available RBA in the data set is determined and stored in the base cluster block.

5. If the file uses a shared resources (LSR) pool, and if the pool is not currently in existence, DFHFCL is called to determine the pool’s characteristics and to build it.
6. Before opening a VSAM file, any STRNO, BUFND, or BUFNI parameters that may have been specified in the JCL DD statement are copied to the FCT entry (for LSR opens, these are ignored). The ACB is now created and its various options and parameters filled in from information in the FCT entry. The OPEN is finally completed by a call to VSAM.
7. If the file refers to a BDAM data set, the assembled DCB is used for the open request and no dynamic setting of DCB options is carried out.
8. After the VSAM file has been successfully opened, certain file attributes are obtained from VSAM and are stored in the FCT entry. These include:

   Key length
   Relative key position
   Base/path/AIX indicator
   KSDS/ESDS/RRDS/VRRDS indicator
   Number of strings required for an update operation.

9. For a file opened for update against a VSAM base data set when the update use count in the DSNB for this data set is zero, the BWO attributes in the ICF catalog are validated to find their current state. This is done by making an INQ_DATASET_STATE call to DFHFCAT, regardless of whether the file is defined in the FCT as eligible for BWO support.

   The file open request is rejected if one of the following is true:

   a. The BWO attributes in the ICF catalog show ***either*** that the data set is “back level”, that is, a backup copy has been restored but not forward recovered, ***or*** that either the catalog or the data set has been corrupted.
   b. The BWO attributes in the FCT entry conflict with those defined in the DSNB, that is, the file has already been opened with different attributes since the DSNB was created.

   If the file is defined in the FCT as eligible for BWO support, the BWO attributes in the ICF catalog are updated by making a SET_BWO_BITS_ENABLED call to DFHFCAT.

However, if the file is not defined in the FCT as eligible for BWO support, but the BWO attributes in the ICF catalog currently show that the VSAM base data set is eligible for BWO support, the BWO attributes in the ICF catalog are disabled by making a SET_BWO_BITS_DISABLED call to DFHFCAT, and CICS issues a warning message.

Note: The ICF BWO attributes are a property of a VSAM sphere; therefore, the VSAM base data set and alternate index path definitions should be consistent. For a general description of the CICS backup while open (BWO) facility, see the *CICS Recovery and Restart Guide*.

10. The base DSNB, and path DSNB if this is a path, are marked as validated and catalogued.

The main actions when processing a close request:

1. If the close request is for the last file that was opened for update against a VSAM base data set and the file is defined in the FCT as eligible for BWO support, the BWO attributes in the ICF catalog are reset so that BWO support is no longer enabled. This is done by making a SET_BWO_BITS_DISABLED call to DFHFCAT.
2. Before performing the access method close for a VSAM file, the number of accumulated EXCPs is obtained by making a call to VSAM and is saved in the FCT entry ready to be sent to the statistics domain as part of the file statistics.
3. A CLOSE request is then made by issuing the appropriate (VSAM or BDAM) macro.
4. The ACB storage is freed, and certain fields in the FCT entry which are no longer valid are cleared.
5. File statistics and data table statistics, if any, are sent to the statistics domain as unsolicited data.
6. If the file being closed uses shared resources, and if it is the last to have been closed in its LSR pool, DFHFCL is called to delete the pool.
7. If the file was dynamically allocated at open time, it is deallocated, leaving a pointer to the DSNAME block in the FCT entry.
8. If the file had an autojournal, then the autojournal log stream is closed.
9. If the base data set was forward recoverable, and its use count is non-zero, then the forward recovery log stream is closed.

How loaded

As a constituent part of DFHFCFS, which is loaded by DFHFCRP as part of file control initialization.

DFHFCNQ (file control non-RLS lock handler)

DFHFCNQ is the file control non-RLS lock handler. It is called using the FCCA RETAIN_DATASET_LOCKS interface to retain locks in cases of backout failure. It is called using the NQNQ INTERPRET_ENQUEUE interface to interpret File Control locks for presentation purposes.

Lock retention

When DFHFCRC encounters a failure during an attempt to backout a unit of work it must retain all record locks held by that UOW for the failing data set. It issues an FCCA RETAIN_DATASET_LOCKS request to DFHFCCA for RLS access data sets and to this DFHFCNQ for non-RLS access data sets.

Lock name interpretation

Non-RLS locks include record locks for all file types, and for VSAM files, mass-insert range locks, load mode locks and ESDS WRITE locks. Each lock belongs to one of some half dozen or so pools created by DFHFCRP during CICS initialization. DFHFCNQ is called using the NQNQ INTERPRET_ENQUEUE interface and is passed the enqueue pool name and the lock identifier. The name of pool to which a lock belongs is sufficient information to allow the identifier to be parsed and its constituents returned to the caller.

The pool names and lock constituents are:

- FCDSRECD - Data set name and record identifier - for VSAM and CICS-maintained data tables
- FCFLRECD - File name and record identifier - for BDAM and user-maintained data tables

- FCDSRNGE - Data set name and record identifier - VSAM range locks
- FCDSLDMD - Data set name - VSAM load mode locks
- FCDSESWR - Data set name - VSAM ESDS WRITE locks
- FCFLUMTL - File name - UMT load locks

DFHFCOR (file control offsite recovery completion)

DFHFCOR is the file control RLS offsite recovery completion transaction.

Transaction CFOR is attached when CICS detects that is has completed its RLS offsite recovery processing. RLS offsite recovery is only performed when OFFSITE=YES is specified as a system initialization override. CFOR may be attached either during RLS warm or emergency restart (if there is no RLS offsite recovery work to be performed) or during file control commit processing (if the commit was for the last remaining item of RLS offsite recovery work).

DFHFCOR issues message DFHFC0575 and awaits an operator reply. When the reply is received, it enables RLS access for new transactions.

DFHFCQI (file control RLS quiesce initiation)

DFHFCQI is the RLS Quiesce Initiation module. It provides code to initiate a quiesce request against a base data set. It also provides code to inquire on the quiesce state of a base data set, and to complete a quiesce request against a base data set. Quiesce initiations are issued by the CICS API, or by CICS internally, or by CICS internally cancelling certain in-progress quiesce operations. Quiesce inquiries are issued via the CICS API. Quiesce completions are issued by CICS internally.

DFHFCQR (file control quiesce receive transaction)

DFHFCQR is the VSAM RLS Quiesce Receive module, running under a dedicated CFQR system transaction. It provides code to take quiesce requests from the CICS VSAM RLS quiesce exit and pass them to DFHFCQU for processing. As DFHFCQR runs under a system transaction, it has full transaction environment which enables it to invoke API-capable global user exits, or to call parts of file control that reference the TCA.

DFHFCQS (file control RLS quiesce send transaction)

DFHFCQS is the VSAM RLS Quiesce Send module, running under a dedicated CFQS system transaction. It provides code to take quiesce requests from another task and pass them to SMSVSAM. As DFHFCQS runs under a system transaction, it has full transaction environment which enables it to invoke API-capable global user exits, or to call parts of file control that reference the TCA. DFHFCQS is called from DFHFCQT, the quiesce system transaction module, if the transaction id under which DFHFCQT was started is 'CFQS'.

DFHFCQT (file control RLS quiesce common system transaction)

DFHFCQT is the file control RLS quiesce common system transaction.

There are two file control system transactions dedicated to RLS quiesce processing: CFQS and CFQR. CFQS sends quiesce requests to SMSVSAM in order to initiate the quiesce or unquiesce of a data set throughout the sysplex. CFQR receives quiesce requests from VSAM RLS and performs the quiesce processing required for the CICS region concerned. These transactions share a common top-level program, DFHFCQT.

There is no DFHFCQT parameter list. The action DFHFCQT takes depends on the transid of the transaction it is running under. If it is CFQS then DFHFCQS SEND_QUIESCES is called. If it is CFQR then DFHFCQR RECEIVE_QUIESCES is called. If DFHFCQS or DFHFCQR subsequently fail with a disastrous error, control is returned to DFHFCQT and a transaction abend is issued, having first re-attached the transaction concerned to ensure that RLS Quiesce support is not lost for ever.

DFHFCQU (file control RLS quiesce processor)

DFHFCQU is the RLS Quiesce Process module. It processes quiesce requests received from SMSVSAM via the quiesce exit mechanism.

DFHFCQX (file control RLS quiesce exit)

DFHFCQX is the RLS Quiesce Exit module. It is called by SMSVSAM whenever the CICS region concerned is required to perform processing for a quiesce request.

The quiesce exit is specified on the RLS control ACB EXLST. The exit simply initiates processing and returns to VSAM. It must not issue any VSAM requests. It is scheduled as an IRB on the TCB that registered the RLS control ACB. Because of the environment DFHFCQX cannot issue CICS requests. GTF tracing is used to trace entry, exit and any errors.In addition, timestamps are made on entry to and exit from DFHFCQX, and are stored in fields FC_DFHFCQX_ENTRY_STCK and FC_DFHFCQX_EXIT_STCK respectively of the File Control Static area.

On entry to DFHFCQX, register 1 contains the address of a VSAM structure mapped by IFGQUIES which defines the quiesce request. The processing of the quiesce request is performed by the CFQR long-running system transaction (DFHFCQR). To communicate the quiesce to CFQR, DFHFCQX creates an FC Quiesce Receive Element (FCQRE) to describe the request, and adds it to a chain in file control static storage, posting an ECB associated with the chain also in FC static.

DFHFCRC (file control recovery control program)

DFHFCRC provides recovery control for file control. All calls from the Recovery Manager domain to file control come through DFHFCRC.

DFHFCRC is called by the Recovery Manager domain to participate in syncpoint and in warm and emergency restart.

Early on during startup File Control registers as a client of the CICS Recovery Manager. During File Control initialization, File Control will add its recovery gate to the kernel, specifying DFHFCRC as the entry point, and then declares the recovery gate to the CICS Recovery Manager via an RMCD SET_GATE call.

At syncpoint, a resource owner such as File Control may be called either

1. to prepare, optionally followed by shunt-unshunt pairs, followed either by calls to backout (as in 2 below) or a call to commit.
2. to backout, which involves start_backout, optional delivery of backout data, and end_backout, followed by prepare and commit, optionally followed by backout retries (which consist of shunt-unshunt pairs followed by the start_backout - delivery of backout data - end_backout - prepare - commit sequence).

At warm or emergency restart, a resource owner such as File Control will be called with start_delivery, optional deliver_recovery and deliver_forget calls, followed by end_deliver.

The Recovery Manager functions processed by DFHFCRC are:
- RMRO PERFORM_PREPARE
- RMRO PERFORM_COMMIT
- RMRO START_BACKOUT
- RMRO DELIVER_BACKOUT_DATA
- RMRO END_BACKOUT
- RMRO PERFORM_SHUNT
- RMRO PERFORM_UNSHUNT
- RMKP TAKE_KEYPOINT
- RMDE START_DELIVERY
- RMDE DELIVER_RECOVERY
- RMDE DELIVER_FORGET

- RMDE END_DELIVERY

DFHFCRC performs different processing depending on the function with which it has been called:

PERFORM_PREPARE

Any active VSAM requests are terminated, and a vote of READ_ONLY is returned if the unit of work did not make any recoverable file control updates, a vote of YES if the prepare was successful, or a vote of NO otherwise.

PERFORM_COMMIT

For a forwards syncpoint, any changes made by the unit of work to recoverable user-maintained data tables are committed. For a backwards syncpoint, locks for any backout-failed data sets are retained. All other locks are released.

On transaction termination, the FLABs and FRAB are freed unless there are FLABs marked for retention. On an intermediate syncpoint, various flags in the FLABs and FRAB are reset to indicate that a commit has been performed.

START_BACKOUT

Any active VSAM requests are terminated, and any changes made by the unit of work to recoverable user-maintained data tables are backed out.

DELIVER_BACKOUT_DATA

The recoverable file control change represented by the log record delivered to DFHFCRC is backed out via calls to DFHFCFR which reverse the update. The change is not backed out if the unit of work has already suffered a backout failure for the data set, or if the data set is in a ’non-RLS update permitted’ state, or if this call is being made as part of a CEMT or EXEC CICS SET DSNAME RESETLOCKS request.

If a failure occurs during the backout, then backout failure processing is carried out.

END_BACKOUT

Under normal conditions there should be no processing required at END_BACKOUT, but it is conceivable that there might be outstanding active VSAM requests to be terminated.

PERFORM_SHUNT

The failed parts of the unit of work's file control structures are put into a condition to survive without an executable transaction environment. This involves retaining any FLABs that are marked for retention, which will allow files to be closed, but not to be reallocated to a different data set.

If this is an intermediate syncpoint, and the shunt is due to a failure in phase 2 of syncpoint, the transactional parts of the unit of work are copied into a new control structure to be passed to the follow-on unit of work. A new FRAB is acquired to anchor this control structure. If this is transaction termination, or the shunt is due to a failure in phase 1 of syncpoint, the transactional parts are cleaned up.

PERFORM_UNSHUNT

The file control structures are converted back into a condition suitable for a unit of work that is in an executable state. Retained FLABs for the unit of work are restored.

TAKE_KEYPOINT

DFHFCRC is called when CICS takes a keypoint, to perform processing required by BWO backup on non-RLS data sets. This involves the writing of a set of ’tie up records’ and the calculation of a new BWO recovery time.

START_DELIVERY

DFHFCIR is called to process the call.

DELIVER_RECOVERY
DFHFCIR is called to process the call.

DELIVER_FORGET
DFHFCIR is called to process the call.

END_DELIVERY
DFHFCIR is called to process the call.

DFHFCRD (file control RLS cleanup transaction)

As soon as CICS detects an SMSVSAM server failure, it runs program DFHFCRD under transaction CSFR to perform cleanup.

Following the server failure all current RLS ACBs become unusable. DFHFCRD scans a chain of files open in RLS mode, which is anchored from file control static storage and call DFHFCFS to perform an IMMEDIATE_CLOSE for each open file.

DFHFCRD then waits:
1. for the last file to close,
2. once the last file has closed, for SMSVSAM to complete any residual requests against the RLS control ACB.

When both these events have occurred, DFHFCRD calls DFHFCCA to perform UNREGISTER_CONTROL_ACB processing in order to clean up the CICS and VSAM state with respect to the control ACB.

DFHFCRD finally posts an ECB which allows dynamic RLS restart to go ahead. Dynamic RLS restart cannot start until DFHFCRD has completed clean up and posted this ECB.

DFHFCRF (file control function shipping interface module)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address
FC_FCRF_ADDRESS stored in FC Static Storage.

Purpose
DFHFCRF is the function shipping interface module. It is called by the access method independent module DFHFCFR for record management requests (e.g. reads, writes, rewrites, etc.) that are to be directed to files that are defined as remote.

DFHFCRF is called with the FCFR parameter list. From this it constructs an FCRF parameter list, which is subsequently passed to DFHISP and, in turn, either to DFHXFX (the MRO transformer) or to DFHXFFP (the ISC transformer).

DFHFCRF executes the following requests from the DFHFCFRR parameter list:
- Simple read requests
  - READ_INTO and READ_SET
- The read update family
  - READ_UPDATE_INTO and READ_UPDATE_SET
  - REWRITE
  - REWRITE_DELETE
  - UNLOCK

- The browse family
  - START_BROWSE
  - RESET_BROWSE
  - READ_NEXT_SET, READ_NEXT_INTO, READ_PREVIOUS_SET, READ_NEXT_UPDATE_SET, READ_NEXT_UPDATE_INTO, READ_PREVIOUS_UPDATE_SET, and READ_PREVIOUS_UPDATE_INTO
  - END_BROWSE
- Write requests
  - WRITE
- Delete requests
  - DELETE

Called by

DFHFCFR, the File Control file request handler.

Inputs

The FCFR parameter list, as defined by the DFHFCFRA DSECT.

Outputs

The FCRF parameter list, as defined by the DFHFCRFA DSECT.

Operation

Traces module entry.

Checks for an explicit SYSID specified on the request and sets the remote system and remote file name in the DFHFCRF parameter list ready for function shipping.

Increments statistics for the type of request.

Checks request specific parameters

Ships the request.

Handles return codes.

Finally, traces the module exit.

How loaded

By FCRP at file control initialization.

DFHFCRL (file control share control block manager)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCRL. The entry point address is held in FC static storage in a field named FC_FCRL_ADDRESS, which is set by DFHFCRP when it loads DFHFCRL.

Purpose

The file control share control block manager is part of the file control component.

This program modifies the CICS specification of a shared resources pool. The changes are allowed to be made only when the actual pool is deleted.

Called by

DFHAMFC, when installing an LSR pool defined by RDO.

Inputs

The FCRL parameter list, as defined by the DFHFCRLA DSECT, is created as part of the subroutine call.

The input parameters are:

Request identifier
Pool identifier
Number of strings
Maximum key length
Share limit
Buffer characteristics

Outputs

The response and reason codes only. These are returned in the FCRL parameter list.

Operation

The SHRCTL block for the specified pool is addressed. A test is made to determine whether or not the pool is currently built; if it is built, the request is rejected with an error response.

The pool characteristics specified in the input parameter list are included in the SHRCTL block.

Finally the information in the SHRCTL block is written to the CICS global catalog.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCRO (file control RLS open/close program)

DFHFCRO performs an equivalent function for RLS opens and closes as is performed by DFHFCN for non-RLS access mode.

DFHFCRP (file control restart program)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCRP. This address is needed only by DFHFCIN2 during initialization; it is therefore not saved in FC static storage.

Purpose

The file control restart program is part of the file control component. This program creates a file control component on a cold or initial start of CICS, or re-creates it after a warm or emergency start. For a warm or emergency start, the intention is to reconstruct the identical file control environment which was in effect at the time of the previous CICS termination.

Called by

DFHFCIN2, during file control initialization.

Inputs

None.

Outputs

The restarted file control component. File control static addresses and indicators are set up. DFHFCRP's response and reason codes are set in the parameter list defined by DFHFCRPA DSECT.

Operation

Calls loader domain to define (if necessary) and acquire (load) the following file control programs: DFHDTINS, DFHFCAT, DFHFCCA, DFHFCDN, DFHFCD2, DFHFCES, DFHFCFL, DFHFCFS, DFHFCIR, DFHFCLF, DFHFCLJ, DFHFCMT, DFHFCNQ, DFHFCQI, DFHFCQU, DFHFCQX, DFHFCRC, DFHFCRL, DFHFCRO, DFHFCRR, DFHFCRS, DFHFCRV, DFHFCSD, DFHFCST, and DFHFCVS.

Adds gates to the kernel for recovery control, ENF services, and log stream failure notification.

Calls storage manager domain to add (create) the following storage subpools: file control general below 16MB, VSAM FCTE, BDAM FCTE, ACB, DCB, SHRCTL, DSN, FFLE, FRAB, FRTE, FLLB, FLAB, RPL, IFGLUWID, file control fixed-length buffer storage. Calls the NQ domain to add (create) enqueue subpools for: dataset record NQs, file record NQs, range NQs, load mode NQs, ESDS write NQs, and UMT loading NQs.

Calls DFHTMP to create TMP primary indexes for the FCT, AFCT, and DSN tables, and a TMP secondary index for the DSN table.

If RLS is supported (correct level of DFSMS, and RLS=YES SIT parameter) initializes the CSFR, CFQS, CFQR and CFOR tasks, registers file control's interest in the SMSVSAM ENF signal by a LISTEN call to DFHDMEN, and calls DFHFCRR to restart RLS.

On a warm or emergency start:

- Determines installation levels of the MVS/Data Facility Product (MVS/DFP) (or DFSMS), the Data Facility Hierarchical Storage Manager (DFHSM), and the Data Facility Data Set Services (DFDSS) for VSAM backup while open (BWO) support.
- Restores DSNAME blocks from the CICS global catalog, recreating a DSN control block in the DSN subpool storage. For each block, adds its DS name to the TMP primary index, and adds its DS number to the TMP secondary index.
- Restores VSAM file entries from the CICS global catalog. For each entry, adds its file name to the TMP FCT index.
- Restores BDAM file entries from the CICS global catalog. For each entry, adds its file name to the TMP FCT index. Further, for each entry, restores the BDAM DCB from the catalog and copies it to an entry in the DCB storage subpool.
- Restores DSNAME references from the CICS global catalog. For each entry, locates its FCTE and invokes DFHFCDN to connect the FCTE to its DSN block.
- Restores SHRCTL blocks from the CICS global catalog.

On a cold start:

- As for a warm or emergency start, determines installation levels of MVS/DFP, DFHSM, and DFDSS for VSAM backup while open (BWO) support.
- Purges the CICS global catalog of all FCTEs, SHRCTL blocks, DSNAME references, AFCTEs, and BDAM DCBs.
- Calls the loader domain to load the FCT specified by the FCT system initialization parameter.
- Builds all eight SHRCTL blocks, using any information that may have been specified in the loaded FCT. Writes the blocks to the CICS global catalog.
- For each file control table entry in the loaded FCT, creates an FCT entry in the FCT storage subpool, copies the information to it, adds the file name to the TMP index, and writes the table entry to the CICS global catalog.
- Calls the loader domain to delete the previously loaded FCT.

Indicates file control restart complete for non-recoverable business by setting FC_NON_REV_ALLOWED_ECB on.

Sends message to inform that file control restart is complete.

If all was successful, turns on the FCSCMPLT flag in FC static.

Finally, posts the FC_RECOV_ALLOWED_ECB in FC static.

How loaded

By the file control initialization module 2, DFHFCIN2, and deleted after it has completed.

DFHFCRR (file control RLS restart)

DFHFCRR is used to restart the RLS component of File Control. It is called whenever CICS is restarted and after any total RLS failure. DFHFCRR is also called whenever a resource can be made available again after earlier failures have been rectified, and after recovery from Lost Locks.

DFHFCRR is invoked whenever CICS is restarted (COLD, WARM or EMERGENCY) by DFHFCRP, and following any total RLS failure (DYNAMIC restart) by DFHFCES.

DFHFCRR is also called to retry work which has been shunted because a resource (a data set, and RLS cache, or the VSAM RLS server) was not available. For this purpose, it is called by DFHFCQU when CICS is notified that a data set has been unquiesced, has completed a non-BWO copy or has completed forward recovery, and when CICS is notified that a previously failed cache is now available; by DFHFCFL when the API interface is used to retry all shunted work for a given data set; and by DFHFCRO when an override condition is detected, in order to drive any shunted work. DFHFCRR is also called by DFHFCQU when CICS is notified that all systems have completed lost locks recovery for a data set.

DFHFCRS (file control RLS record management processor)

DFHFCRS performs an equivalent function for RLS access mode record management requests as is performed by DFHFCVS for non-RLS access mode requests.

DFHFCRV (file control RLS VSAM interface processor)

DFHFCRV performs an equivalent function for RLS access mode record management requests as is performed by DFHFCVR for non-RLS access mode requests.

DFHFCSD (file control shutdown program)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCSD. The entry point address is held in FC static storage in a field named FC_FCSD_ADDRESS, which is set by DFHFCRP when it loads DFHFCSD.

Purpose

The file control shutdown program is part of the file control component. Its purpose is to close all CICS files that are still open during phase 2 of a normal controlled CICS termination. This processing is bypassed for immediate termination.

Called by

DFHSTP, to close all open files managed by CICS file control.

Inputs

The FCSD parameter list, as defined by the DFHFCSDA DSECT, is created as part of the subroutine call.

The input parameters are:

Type of shutdown (immediate, warm)

Outputs

The response and reason codes only, which are returned in the FCSD parameter list.

Operation

DFHFCSD has only one function: TERMINATE.

On a 'warm' shutdown (that is, a not-immediate shutdown), DFHFCSD calls DFHTMP to scan all FCT entries. For each file, it calls DFHFCFS to close the file. A special CLOSE qualifier (shutdown) is specified on the call to DFHFCFS so as not to catalog the FCT entry as in an 'unenabled' state. DFHFCSD also calls DFHFCDO to disconnect coupling facility data table pools.

If RLS is supported, the quiesce system tasks CFQS and CFQR are terminated.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCST (file control statistics program)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCST. The entry point address is held in FC static storage in a field named FC_FCST_ADDRESS, which is set by DFHFCRP when it loads DFHFCST.

Purpose

The file control statistics program is part of the file control component.

This program is called to collect statistics for a single file, together with any data table statistics, or to collect statistics for the activity in a shared resources pool.

It is also called to return file statistics associated with a file's use of a shared resources pool.

Called by

DFHSTFC	Collect file statistics
DFHSTLS	Collect pool statistics and also file-in-pool statistics.

Inputs

The FCST parameter list, as defined by the DFHFCSTA DSECT, is created as part of the subroutine call.

The input parameters are:

Request identifier
File name
FCTE token
Statistics record
Pool identifier
Browse token
Reset indicator

Outputs

Returned in the FCST parameter list:

Browse token
Response
Reason

Operation

- Collect file statistics:

  The FCT entry token is validated if supplied; otherwise, the file name is used to locate the FCT entry.

  The file statistics, and any data table statistics, are collected from the FCTE and copied into the statistics record. The statistics in the FCTE are optionally reset according to the reset indicator.

  For data tables, a STATISTICS data table service request is issued to retrieve and reset those statistics that are maintained by data table services. These statistics are appended to the file statistics record.

  The FCT entry is unlocked and the statistics record returned to the caller.
- Collect pool statistics:

  The SHRCTL block for the specified pool is addressed. The pool statistics are copied into the statistics record and are returned to the caller.
- Start browse of files in pool:

  Storage is obtained from the general file control pool for the browse cursor. The browse token is returned to the caller.
- Get statistics for next file in pool:

  DFHTMP is invoked to locate the FCT entry identified by the browse cursor. If the file uses the specified pool, the shared pool statistics for this file are retrieved and returned in the statistics record.

  The statistics contain the data and index buffer sizes, and the number of times buffer waits occurred.

  The browse cursor is updated before returning to the caller.
- End browse of files in pool:

  The browse cursor storage is freed before returning to the caller.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCVR (file control VSAM interface program)

Call mechanism

BALR, obtaining LIFO storage on entry.

Entry address

DFHFCVR. DFHFCVR is link-edited with DFHFCVS. For calls to DFHFCVR from DFHFCVS, the entry point address is known to DFHFCVS from the link-edit. This address is also stored in FC static storage in a field named FC_FCVR_ENTRY. In addition, there is a further “entry address”, UPADEXIT, which is the entry code for the UPAD exit code.

Purpose

The VSAM request interface program is part of the file control component.

This module contains code that issues the VSAM requests, and performs UPAD exit processing in the case of synchronous requests to LSR files, or performs the IOEVENT wait (‘FCIOWAIT’) in the case of asynchronous requests to NSR files.

The module also contains a number of further routines that implement functions required by DFHFCVS.

Called by

DFHFCBD	To issue a message
DFHFCFR	To wait on a CICS ECB
DFHFCVR	Recursively, in order to issue an ENDREQ request to free a deadlock
DFHFCVS	When issuing VSAM requests
DFHFCVS	To execute one of the constituent functions
VSAM	To invoke the UPAD exit.

Inputs

The FCWSV parameter list, as defined by the DFHFCWS macro, is created in the caller's automatic storage and addressed by register 1 on the call. The input parameters are:

Request identifier
FCTE address
VSWA address
ECB address
Wait resource type
Message number
Dump code

In addition, DFHFCVR requires access to the TCA for certain of its operations.

Outputs

FCVR_RESPONSE parameter (only), defined as part of the FCWSV parameter list.

Operation

Initialize: Copies the VSAM exit list to FC static storage. This action is performed as part of file control initialization.

VSAM_Request: Issues the request to VSAM. Performs the IOEVENT wait. Handles LSR 'no buffers' logical error. Issues change mode request to perform the request under the concurrent TCB if possible.

Get_Strings and Free_Strings: Acquires and frees the required number of shared strings from the LSR pool.

Get_TRANID and Free_TRANID: Allocates and releases a VSAM tranid required during sequential update operations to an LSR file.

Wait_CICSECB: Issues a function request to wait for a CICS ECB to be posted.

Wait_String: Issues a function request to wait for a private string to become available.

Send_Message: Issues a function request to send a message.

How loaded

Link-edited with DFHFCVS to form the DFHFCVS load module, which is loaded by DFHFCRP as part of file control initialization.

DFHFCVS (file control VSAM request processor)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCVS. The entry point address is held in FC static storage in a field named FC_FCVS_ADDRESS, which is set by DFHFCRP when it loads DFHFCVS.

Purpose

Processes file control requests to VSAM files.

Also initializes certain FC static storage fields during file control initialization.

Called by

DFHFCDTS To access the VSAM source data set to satisfy requests that cannot be satisfied by the table itself

DFHFCFR After having determined that the request is for a VSAM file.

Inputs

The FCFR parameter list, as defined by the DFHFCFRA DSECT. Also the file control environment, including FC static storage and the FCT.

Outputs

Updated FCFR parameter list.

Operation

Selects on the request type, and passes control to the routine specific to that request.

Acquires and releases the VSWA as necessary.

Logs and journals the request if required.

Performs record-length and key-length checking.

Acquires storage, in the correct key subpool, for requests that specify SET.

Calls DFHFCVR to perform the VSAM request.

Resolves conflicts of exclusive control.

Performs record locking and resolves locking conflicts, including the detection of deadlocks caused either by single tasks that deadlock themselves or by multiple tasks that deadlock each other.

Performs initialization of FC static storage during file control initialization.

For CICS-maintained data tables, calls data table services to update the table to keep it in step with the VSAM source data set.

How loaded

By DFHFCRP as part of file control initialization.

Parameter lists

File control provides the following functions in OCO modules:

FCCA CHECK function

CHECK is issued to get the results of a previous, asynchronous, operation.

Input parameters

CHECK_TOKEN is a token that was returned on the previous request for which the results are being checked.

Output parameters

ACCMETH_RETURN_CODE
: is a two-byte code returned by SMSVSAM.

CONFLICTING_QUIESCE
: indicates the type of quiesce which conflicts with this request, and can have any of these values:

`QUIESCE|UNQUIESCE|NONBWO_END|BWO_END|NONBWO_START|`
`BWO_START`

RESPONSE
: is DFHFCCA's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
: is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	VSAM_REQUEST_ERROR RLS_FAILURE
DISASTER	ABEND

FCCA COLD_START_RLS function

This request is issued as part of CICS cold start processing. CICS issues an IDARECOV TYPE=COLDSTART call to SMSVSAM to release all RLS locks owned by this CICS, and to clear the lost locks status and 'non RLS update permitted' state of all data sets with respect to this CICS.

Input parameters

SUBSYSNM
: is a pointer to an IFGSYSNM structure.

Output parameters

ACCMETH_RETURN_CODE
: is a two-byte code returned by SMSVSAM.

RESPONSE
: is DFHFCCA's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
: is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	VSAM_REQUEST_ERROR RLS_FAILURE
DISASTER	ABEND

FCCA DRAIN_CONTROL_ACB function

The Control ACB must be drained when file control detects that an instance of the SMSVSAM server has become failed. DFHFCCA will set an indicator in FC static storage so that no other RLS activity may proceed, and then drain all existing RLS access. This involves incrementing the server sequence number in FC static storage, closing all RLS ACBs and unregistering the Control ACB.

Input parameters

None

Output parameters

RESPONSE
: is DFHFCCA's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
: is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DISASTER_PERCOLATION ABEND

FCCA INQUIRE_RECOVERY function

This request is issued as part of CICS start up processing. CICS makes an IDAINQRC request to VSAM to obtain the information necessary to determine what RLS recovery actions are required by CICS.

Input parameters

AREA_PTR
is the address of an area in which the IFGINQRC information is to be returned.

AREA_LENGTH
is the length of the supplied area.

Output parameters

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

REQUIRED_LENGTH
is the length of the IFGINQRC area to be returned, if it exceeds the length of the supplied area.

RESPONSE
is DFHFCCA's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	AREA_TOO_SMALL VSAM_REQUEST_ERROR RLS_FAILURE
DISASTER	ABEND

FCCA LOST_LOCKS_COMPLETE function

CICS issues an IDARECOV TYPE=LL request to SMSVSAM when it has completed recovery processing for a data set that is in lost locks status. SMSVSAM resets the state of the data set in the sharing control data set to indicate that the data set is no longer lost locks with respect to this CICS.

Input parameters

DATASET
is the 44-character name of the base data set for which CICS has completed lost locks recovery.

[RESTART]
is an optional parameter which indicates whether the call was issued by file control restart. It can have either of these values:

YES|NO

Output parameters

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

RESPONSE
is DFHFCCA's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	VSAM_REQUEST_ERROR RLS_FAILURE
DISASTER	ABEND

FCCA QUIESCE_COMPLETE function

When CICS has completed the processing required by it for a quiesce request from SMSVSAM, it issues an IDAQUIES call to SMSVSAM with a quiesce type of QUICMP.

Input parameters

DATASET is the 44-character name of the base data set for which quiesce processing is complete.

VSAM_QUIESCE_TOKEN
is a token used to relate quiesce completion to the quiesce request which has been completed, and which is supplied by SMSVSAM when the quiesce request is received by CICS.

Output parameters

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

RESPONSE is DFHFCCA's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	VSAM_REQUEST_ERROR RLS_FAILURE
DISASTER	ABEND

FCCA QUIESCE_REQUEST function

DFHFCCA issues quiesce requests to SMSVSAM on behalf of the quiesce component of CICS. It issues IDAQUIES calls of the following types:

- QUICLOSE to request SMSVSAM to notify all CICS systems that have ACBs open against this data set that these ACBs are to be closed. In addition the data set is marked in the VSAM catalog as being quiesced once these ACBs have been closed.
- QUIOPEN to request SMSVSAM to mark the data set as no longer quiesced, i.e. unquiesced. In addition QUIOPEN will cancel an in-progress QUICLOSE.
- QUIBEND to request SMSVSAM to cancel an in-progress BWO backup of a data set.
- QUICEND to request SMSVSAM to cancel an in-progress non-BWO backup of a data set.

Input parameters

DATASET is the 44-character name of the base data set to be quiesced.

QUIESCE_TYPE is the type of quiesce, and can have any of these values:

QUIESCE|UNQUIESCE|NONBWO_END|BWO_END

QUIESCE_TYPE is the type of quiesce, and can have any of these values:

QUIESCE|UNQUIESCE|NONBWO_END|BWO_END

[IMMEDIATE] applies only when the quiesce type is quiesce, and indicates whether or not the quiesce will force files to close immediately, or will allow in-flight units of work to reach syncpoint. It can have either of these values:

YES|NO

Output parameters

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

CHECK_TOKEN is a token which will be used on the CHECK request.

CONFLICTING_QUIESCE
indicates the type of quiesce which conflicts with this request, and can have any of these values:

QUIESCE|UNQUIESCE|NONBWO_END|BWO_END|NONBWO_START|
BWO_START

RESPONSE is DFHFCCA's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	VSAM_REQUEST_ERROR RLS_FAILURE
DISASTER	ABEND

FCCA REGISTER_CONTROL_ACB function

The Control ACB is 'opened' using an IDAREGP request to SMSVSAM. The Control ACB must be registered before CICS can open any 'ordinary' ACB for RLS access.

Input parameters

None

Output parameters

VSAM_RETURN_CODE
is a fullword return code from VSAM.

VSAM_REASON_CODE
is a fullword return code from VSAM.

VSAM_ERROR_DATA
is an 8-byte field containing error data returned by VSAM.

RESPONSE
is DFHFCCA's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	RLS_FAILURE VSAM_REQUEST_ERROR
DISASTER	DISASTER_PERCOLATION ABEND

FCCA RELEASE_LOCKS function

CICS issues an IDALKREL request to SMSVSAM as part of commit processing at the end of every unit of work. It requests VSAM to release all locks owned by the unit of work.

Input parameters

LUWID
is a pointer to an IFGLUWID structure containing the id for the unit of work.

[RESTART]
is an optional parameter which indicates whether the call was issued by file control restart. It can have either of these values:

`YES|NO`

Output parameters

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

RESPONSE
is DFHFCCA's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	VSAM_REQUEST_ERROR RLS_FAILURE
DISASTER	ABEND

FCCA RESET_NONRLS_BATCH function

CICS issues an IDARECOV TYPE=NONRLS request to VSAM when it has completed processing the NSR batch override response from an RLS file open. SMSVSAM resets the state of the data set in the sharing control data set to indicate that the batch override (or 'non RLS update permitted') state no longer needs to be reported to this CICS when it opens the data set.

Input parameters

DATASET
is the 44-character name of the base data set for which the state is to be cleared.

Output parameters

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

RESPONSE
is DFHFCCA's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	VSAM_REQUEST_ERROR RLS_FAILURE
DISASTER	ABEND

FCCA RETAIN_DATASET_LOCKS function

CICS issues an IDARETLK TYPE=SS call to SMSVSAM when a unit of work has suffered a backout failure on a data set. This requests SMSVSAM to mark all locks against the data set owned by the unit of work for conversion into retained locks on a subsequent IDALKREL call.

Input parameters

LUWID
is a pointer to an IFGLUWID structure containing the id for the unit of work.

DATASET
is the 44-character name of the base data set which has suffered a backout failure.

Output parameters

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

RESPONSE
is DFHFCCA's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	VSAM_REQUEST_ERROR RLS_FAILURE
DISASTER	ABEND

FCCA RETAIN_UOW_LOCKS function

CICS issues an IDARETLK TYPE=IND call to SMSVSAM when a unit of work has encountered an in-doubt failure. This requests VSAM to mark all locks owned by the unit of work for conversion into retained locks on a subsequent IDALKREL call.

Input parameters

LUWID
is a pointer to an IFGLUWID structure containing the id for the unit of work.

Output parameters

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

RESPONSE
is DFHFCCA's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	VSAM_REQUEST_ERROR RLS_FAILURE
DISASTER	ABEND

FCCA UNREGISTER_CONTROL_ACB function

The RLS Control ACB is 'closed' using an IDAUNRP request to SMSVSAM. The Control ACB cannot be unregistered while there are any 'ordinary' ACBs open for RLS access.

Input parameters

None

Output parameters

VSAM_RETURN_CODE
is a fullword return code from VSAM.

VSAM_REASON_CODE
is a fullword reason code from VSAM.

RESPONSE
is DFHFCCA's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	RLS_FAILURE VSAM_REQUEST_ERROR
DISASTER	DISASTER_PERCOLATION ABEND

FCCI INQUIRE function

FCCI is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for the table inquire function. It is not used by CICS.

FCCR POINT function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The POINT function locates a record in a coupling facility data table.

Input parameters

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY
is the 16-byte key of the record to be accessed. For approximate key operations, this specifies the start key and is updated on successful completion to contain the key of the record actually accessed.

KEY_COMPARISON
is the comparison condition, and can take the values

LT|LTEQ|EQ|GTEQ|GT

KEY_MATCH_LENGTH
is the key match length for generic key operations.

UOW_ID
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

KEY returns the 16-byte key of the located record.

RESPONSE is DFHFCCR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND TABLE_LOADING TABLE_TOKEN_INVALID TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR HIGHEST function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The HIGHEST function returns the highest key in a coupling facility data table, if any.

Input parameters

TABLE_NAME is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN is the token returned on OPEN which must be passed on all subsequent requests against that open table.

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

KEY returns the 16-byte key of the highest record.

RESPONSE is DFHFCCR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND TABLE_LOADING TABLE_TOKEN_INVALID TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR READ function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The READ function reads a record from a coupling facility data table, optionally for update.

Input parameters

TABLE_NAME is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY_COMPARISON
is the comparison condition, and can take the values

LT|LTEQ|EQ|GTEQ|GT

KEY_MATCH_LENGTH
is the key match length for generic key operations.

KEY
is the 16-byte key of the record to be accessed. For approximate key operations, this specifies the start key and is updated on successful completion to contain the key of the record actually accessed.

BUFFER
is the input buffer for read requests.

UOW_ID
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

SUSPEND
specifies whether to wait if the requested record is locked by an active lock, and can take the values

YES|NO

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

UPDATE_TOKEN
returns a token on a read for update.

KEY
returns the 16-byte key of the highest record.

LOCK_OWNER_SYSTEM
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_APPLID
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

RESPONSE
is DFHFCCR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND RECORD_BUSY RECORD_LOCKED TABLE_LOADING INVALID_REQUEST INCOMPLETE_UPDATE TABLE_TOKEN_INVALID TABLE_DESTROYED UOW_FAILED UOW_NOT_IN_FLIGHT UOW_TOO_LARGE POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR READ_DELETE function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The READ_DELETE function reads and deletes a record from a coupling facility data table. It is not used by CICS.

FCCR UNLOCK function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The UNLOCK function unlocks a record previously read for update in a coupling facility data table.

Input parameters

TABLE_NAME is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY is the 16-byte key of the record to be unlocked.

BUFFER is the input buffer for read requests.

UPDATE_TOKEN is the token returned on the preceding read for update.

UOW_ID is the unit of work identification, which is required for the locking model (non-recoverable or recoverable).

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

RESPONSE is DFHFCCR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND RECORD_CHANGED TABLE_LOADING INVALID_REQUEST UPDATE_TOKEN_INVALID TABLE_TOKEN_INVALID TABLE_DESTROYED UOW_FAILED UOW_NOT_IN_FLIGHT POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR LOAD function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The LOAD function adds a record to a coupling facility data table during loading.

Input parameters

TABLE_NAME is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY is the 16-byte key of the record to be loaded.

DATA is the address and length of the record data to be loaded.

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

RESPONSE is DFHFCCR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED DUPLICATE_RECORD MAXIMUM_RECORDS_REACHED NO_SPACE_IN_POOL INVALID_REQUEST INVALID_LENGTH TABLE_TOKEN_INVALID TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR WRITE function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The WRITE function writes a new record to a coupling facility data table.

Input parameters

TABLE_NAME is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY is the 16-byte key of the record to be added.

DATA is the address and length of the record data to be added.

UOW_ID is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

SUSPEND specifies whether to wait if the requested record is locked by an active lock, and can take the values

YES|NO

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

LOCK_OWNER_SYSTEM
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_APPLID
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

RESPONSE is DFHFCCR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED DUPLICATE_RECORD RECORD_BUSY RECORD_LOCKED MAXIMUM_RECORDS_REACHED NO_SPACE_IN_POOL TABLE_LOADING INVALID_REQUEST INVALID_LENGTH UPDATE_TOKEN_INVALID INCOMPLETE_UPDATE TABLE_TOKEN_INVALID TABLE_DESTROYED UOW_FAILED UOW_NOT_IN_FLIGHT UOW_TOO_LARGE POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR REWRITE function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The REWRITE function rewrites an an existing record in a coupling facility data table, following a read for update.

Input parameters

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY
is the 16-byte key of the record to be rewritten.

DATA
is the address and length of the record data to be rewritten.

UPDATE_TOKEN
is the token returned on the preceding read for update.

UOW_ID
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

SUSPEND
specifies whether to wait if the requested record is locked by an active lock, and can take the values

YES|NO

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

LOCK_OWNER_SYSTEM
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_APPLID
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

RESPONSE
is DFHCCR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND RECORD_CHANGED RECORD_BUSY RECORD_LOCKED MAXIMUM_RECORDS_REACHED NO_SPACE_IN_POOL TABLE_LOADING INVALID_REQUEST INVALID_LENGTH UPDATE_TOKEN_INVALID INCOMPLETE_UPDATE TABLE_TOKEN_INVALID TABLE_DESTROYED UOW_FAILED UOW_NOT_IN_FLIGHT UOW_TOO_LARGE POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR DELETE function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The DELETE function deletes a record from a coupling facility data table, following a read for update.

Input parameters

TABLE_NAME is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY_COMPARISON
is the comparison condition, and can take the values

LT|LTEQ|EQ|GTEQ|GT

KEY_MATCH_LENGTH
is the key match length for generic key operations.

KEY is the 16-byte key of the record to be deleted.

UPDATE_TOKEN is the token returned on the preceding read for update.

UOW_ID is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

SUSPEND specifies whether to wait if the requested record is locked by an active lock, and can take the values

YES|NO

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

KEY is the 16-byte key of the record actually deleted.

LOCK_OWNER_SYSTEM
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_APPLID
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

RESPONSE is DFHFCCR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND RECORD_CHANGED RECORD_BUSY RECORD_LOCKED TABLE_LOADING INVALID_REQUEST UPDATE_TOKEN_INVALID INCOMPLETE_UPDATE TABLE_TOKEN_INVALID TABLE_DESTROYED UOW_FAILED UOW_NOT_IN_FLIGHT UOW_TOO_LARGE POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR DELETE_MULTIPLE function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The DELETE_MULTIPLE function deletes records from a coupling facility data table, subject to key match conditions, until no more records match or an exception occurs.

Input parameters

TABLE_NAME is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY_COMPARISON
is the comparison condition, and can take the values

LT|LTEQ|EQ|GTEQ|GT

KEY_MATCH_LENGTH
is the key match length for generic key operations.

KEY is the 16-byte key of the record(s) to be deleted.

UOW_ID is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

SUSPEND specifies whether to wait if the requested record is locked by an active lock, and can take the values

YES|NO

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

DELETED_RECORD_COUNT
is the number of records successfully deleted by the delete_multiple request.

KEY is the 16-byte key of the last record deleted.

LOCK_OWNER_SYSTEM
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_APPLID
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

RESPONSE
is DFHFCCR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND RECORD_CHANGED RECORD_BUSY RECORD_LOCKED TABLE_LOADING INVALID_REQUEST UPDATE_TOKEN_INVALID INCOMPLETE_UPDATE TABLE_TOKEN_INVALID TABLE_DESTROYED UOW_FAILED UOW_NOT_IN_FLIGHT UOW_TOO_LARGE POOL_STATE_ERROR CF_ACCESS_ERROR

FCCT OPEN function

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The OPEN function defines a coupling facility data table table and establishes a connection between it and a CICS file. A security check is performed for access to the table name. If the table does not exist, it is implicitly created. If the table requires loading, it can only be opened if the access mode specifies exclusive access (or prefer_shared, allowing exclusive access if necessary).

Input parameters

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

RECORD_LENGTH
specifies the maximum record length, in the range 1 to 32767.

KEY_LENGTH
specifies the key length, in the range 1 to 16.

MAXIMUM_RECORDS
specifies the maximum number of records which can be stored in the table.

UPDATE_MODEL
specifies the method to be used for updating. It can take any of the values:

CONTENTION|LOCKING|RECOVERABLE

Contention means version compare and swap. Locking means normal update locking. Recoverable includes backout support in addition to the basic locking model.

INITIAL_LOAD
specifies whether initial load is required. It can take the values:

YES|NO

OPEN_MODE
specifies a read_only or read_write open. It can take the values

READ_ONLY|READ_WRITE

ACCESS_MODE
specifies whether the table is being opened for exclusive or shared use. It can take the values:

EXCLUSIVE|SHARED|PREFER_SHARED

Only one user at a time can have an exclusive open active. If the table requires loading and is not yet being loaded, it can only be opened in exclusive mode. If

PREFER_SHARED is specified, the table will be opened in exclusive mode if loading is required, otherwise it will be open in shared mode.

SHARED_ACCESS
specifies for an exclusive mode open whether other users will be allowed shared access to the file at the same time. It can take the values:

NONE|READ_ONLY|READ_WRITE

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

TABLE_TOKEN is a unique token representing the connection to this table. It must be passed on all subsequent requests against that open table, including close and set.

RECORD_LENGTH
returns the maximum record length of the table.

KEY_LENGTH returns the key length of the table.

MAXIMUM_RECORDS
returns the maximum number of records limit for the table.

UPDATE_MODEL returns the update model for the data table. It can take any of the values:

CONTENTION|LOCKING|RECOVERABLE

Contention means version compare and swap. Locking means normal update locking. Recoverable includes backout support in addition to the basic locking model.

INITIAL_LOAD returns whether or not the data table requires initial loading. It can take the values:

YES|NO

ACCESS_MODE returns whether the table was opened for exclusive or shared use. It can take the values:

EXCLUSIVE|SHARED

LOADED returns an indication of whether the table has been loaded. If the table was created as empty this is set to yes as if loading were already done. It can take the values:

YES|NO

CURRENT_USERS
returns the number of explicit opens which are currently active against the table (not including internal recoverable opens issued by the server).

CURRENT_RECORDS
returns the number of records in the data table.

CURRENT_HIGH_KEY
returns the key of the last record in the table at the time of the request, or low values if the table does not contain any records.

RESPONSE is DFHFCCT's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED ACCESS_NOT_ALLOWED TABLE_NOT AVAILABLE NOT_YET_LOADED SHARED_ACCESS_CONFLICT EXCLUSIVE_ACCESS_CONFLICT INCOMPATIBLE_ATTRIBUTES INCOMPLETE_ATTRIBUTES INCORRECT_STATE RECOVERY_NOT_ENABLED OPTION_NOT_SUPPORTED NO_SPACE_IN_POOL MAXIMUM TABLES_REACHED TOO_MANY_USERS TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCT CLOSE function

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The CLOSE function terminates the connection to the specified table.

Input parameters

TABLE_NAME is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN is the token which was returned by the open.

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

RESPONSE is DFHFCCT's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED TABLE_TOKEN_INVALID TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCT DELETE function

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The DELETE function deletes a coupling facility data table, provided that it is not currently open. A security check for table access is performed.

Input parameters

TABLE_NAME is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

RESPONSE is DFHFCCT's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED ACCESS_NOT_ALLOWED TABLE_NOT_FOUND EXCLUSIVE_ACCESS_CONFLICT TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCT SET function

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The SET function is used to change the attributes of a table. The maximum number of records can be changed, the open mode can be changed to indicate no longer loading, and the access mode can be changed from exclusive to shared.

Input parameters

TABLE_NAME is the 16-character name of the CFDT (8 characters padded with trailing spaces).

MAXIMUM_RECORDS
specifies the maximum number of records which can be stored in the table.

AVAILABLE indicates whether new open requests are to be allowed for this table. It can take the values:

YES|NO

LOADED indicates whether the table is to be marked as loaded. It can take the values:

YES|NO

ACCESS_MODE specifies the access mode which is to be set for the table. It can take the values:

EXCLUSIVE|SHARED

The access mode is normally set to shared when a data table load has completed.

SHARED_ACCESS
specifies the shared access which is to be allowed by other users when the access mode is shared.

NONE|READ_ONLY|READ_WRITE

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

RESPONSE is DFHFCCT’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED ACCESS_NOT_ALLOWED TABLE_NOT_FOUND SHARED_ACCESS_CONFLICT EXCLUSIVE_ACCESS_CONFLICT ALREADY_SET INCORRECT_STATE OPTION_NOT_SUPPORTED TABLE_TOKEN_INVALID TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCT EXTRACT_STATISTICS function

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The EXTRACT_STATISTICS function returns information about a table which is currently open, with optional reset.

Input parameters

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token which was returned by the open.

RESET_STATISTICS
is an optional parameter which specifies whether or not statistics are to be reset. It can take the values

YES|NO

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

CURRENT_USERS
is the number of explicit opens which are currently active against the table (not including internal recoverable opens issued by the server).

CURRENT_RECORDS
is the number of records currently in the data table.

HIGHEST_RECORDS
is the highest number of records in the table as seen by the current server at any time since the last statistics reset.

CONTENTION_COUNT
is the number of times a rewrite or delete failed because of a mismatched version (for the contention model) or the number of times that a lock was found to be unavailable (for the locking or recoverable models) since the last statistics reset.

RESPONSE
is DFHFCCT's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED TABLE_TOKEN_INVALID

FCCU PREPARE function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The PREPARE function prepares to commit a unit of work.

Input parameters

UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.

TRANSACTION_NUMBER
is used for debug trace purposes.

Output parameters

RESPONSE
is DFHFCCU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES UOW_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR

FCCU RETAIN function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The RETAIN function marks a unit of work as retained.

Input parameters

UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.

TRANSACTION_NUMBER
is used for debug trace purposes.

Output parameters

RESPONSE
is DFHFCCU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES UOW_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR

FCCU COMMIT function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The COMMIT function commits a unit of work.

Input parameters

UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.

TRANSACTION_NUMBER
is used for debug trace purposes.

Output parameters

RESPONSE
is DFHFCCU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES UOW_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR

FCCU BACKOUT function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The BACKOUT function backs out a unit of work.

Input parameters

UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.

TRANSACTION_NUMBER
is used for debug trace purposes.

Output parameters

RESPONSE
is DFHFCCU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES POOL_STATE_ERROR CF_ACCESS_ERROR

FCCU INQUIRE function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The INQUIRE function inquires about the status of a unit of work.

Input parameters

UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.

UOW_RESTARTED
is an optional parameter which indicates whether the inquire should select only units of work which have been through restart processing, and can take the values:

NO|YES

TRANSACTION_NUMBER
is used for debug trace purposes.

BROWSE
specifies whether the inquire is for a single unit of work or for the first or next UOW in a browse. If omitted, a single UOW inquire is performed. If specified, it can take the values

FIRST|NEXT

FIRST indicates a search for a UOWID greater than or equal to the specified UOWID, and NEXT indicates a search for a UOWID greater than the specified UOWID.

Output parameters

UOW_STATE
indicates the state of an active unit of work, and can have any of the values:

IN_FLIGHT|IN_DOUBT|IN_COMMIT|IN_BACKOUT

In_flight means that the unit of work has made some changes but has not yet reached the stage of prepare to commit. In_doubt means that it has been prepared but not committed or backed out. In_commit means that commit processing has been started. In_backout means that backout processing has been started. (When commit or backout processing completes, the unit of work is deleted).

UOW_ID
is the CICS unit of work id of the UOW for which inquire data is being returned.

UOW_RESTARTED
indicates whether the unit of work has been through restart processing, and can take the values:

NO|YES

UOW_RETAINED
indicates whether the locks for the unit of work have been marked as retained, either explicitly within the current connection or implicitly by a restart. It can take the values:

NO|YES

RESPONSE
is DFHFCCU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND

FCCU RESTART function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The RESTART function establishes recovery status on connecting to a CFDT server.

Input parameters

UOW_SUBSYSTEM_NAME
is not specified by CICS (the CICS applid is used by default).

TRANSACTION_NUMBER
is used for debug trace purposes.

Output parameters

RESPONSE
is DFHFCCU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED SUBSYSTEM_ALREADY_ACTIVE RESTART_ALREADY_ACTIVE TABLE_OPEN_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR

FCDS EXTRACT_CFDT_STATS function

This function causes statistics relating to coupling facility data table usage to be extracted from the coupling facility data tables server.

Input parameters

FCTE_POINTER is the address of the FCTE entry of the file for which CFDT statistics are to be extracted.

RESET_STATISTICS
indicates whether the statistics fields are to be reset to zero or not. It takes the values

YES|NO

TRANSACTION_NUMBER
is an optional parameter which allows the transaction number to be passed to the CFDT server for inclusion in trace messages.

Output parameters

CURRENT_USERS
is an optional fullword parameter which returns the current number of users of the coupling facility data table (that is, the number of opens issued against it).

MAXIMUM_RECORDS
is an optional fullword parameter which returns the current value of the MAXNUMRECS limit for the data table.

CURRENT_RECORDS
is an optional fullword parameter which returns the current number of records in the coupling facility data table.

HIGHEST_RECORDS
is an optional fullword parameter which returns the highest number of records which have ever been in this coupling facility data table since it was last created.

CONTENTION_COUNT
is an optional fullword parameter which returns the number of contentions which have been detected, for a coupling facility data table which uses the contention update model.

RESPONSE is DFHFCDS's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR CFDT_REOPEN_ERROR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_STATS_ERROR CFDT_SYSIDERR CFDT_TABLE_GONE
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	POOL_ELEMENT_NOT_FOUND ABEND DISASTER_PERCOLATION

FCDS DISCONNECT_CFDT_POOLS function

This function causes CICS to disconnect from any coupling facility data table pools to which it is connected.

Input parameters

None

Output parameters

RESPONSE
is DFHFCDS's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CFDT_DISCONNECT_ERROR
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDU PREPARE function

This function causes the coupling facility data table server to be called to prepare a unit of work which has made recoverable updates to one or more coupling facility data tables.

Input parameters

POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for which the prepare is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The prepare call will be issued to the CFDT server for this pool.

POOL_NAME
is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

UOW_ID
is the identifier for the unit of work which is to be prepared.

Output parameters

RESPONSE
is DFHFCDU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES UOW_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR CFDT_SYSIDERR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR RESYNC_RETRY_FAILED

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDU RETAIN function

This function causes the coupling facility data table server to be called to convert locks held by the unit of work against recoverable coupling facility data tables into retained locks.

Input parameters

POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for which the retain is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The retain call will be issued to the CFDT server for this pool.

POOL_NAME
is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

UOW_ID
is the identifier for the unit of work for which locks are to be retained.

Output parameters

RESPONSE
is DFHFCDU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES UOW_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR CFDT_SYSIDERR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR RESYNC_RETRY_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDU COMMIT function

This function causes the coupling facility data table server to be called to commit a unit of work which has made recoverable updates to one or more coupling facility data tables.

Input parameters

POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for which the commit is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The commit call will be issued to the CFDT server for this pool.

POOL_NAME is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

UOW_ID is the identifier for the unit of work which is to be committed.

Output parameters

RESPONSE is DFHFCDU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES UOW_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR CFDT_SYSIDERR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR RESYNC_RETRY_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDU BACKOUT function

This function causes the coupling facility data table server to be called to backout a unit of work which has made recoverable updates to one or more coupling facility data tables.

Input parameters

POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for which the backout is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The backout call will be issued to the CFDT server for this pool.

POOL_NAME is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

UOW_ID is the identifier for the unit of work which is to be backed out.

Output parameters

RESPONSE is DFHFCDU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES POOL_STATE_ERROR CF_ACCESS_ERROR CFDT_SYSIDERR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR RESYNC_RETRY_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDU INQUIRE function

This function causes an INQUIRE to be issued to the coupling facility data table in order to obtain information about the status of an active unit of work. If the BROWSE parameter is specified, then the function will return the status of the next unit of work in the browse.

Input parameters

POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for which the INQUIRE is to be issued. The inquire call will be issued to the CFDT server for this pool.

POOL_NAME
is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

UOW_ID
identifies the unit of work for which status information is to be returned, or gives the previous unit of work in the browse.

UOW_RESTARTED
is an optional input parameter which indicates whether or not the inquire should select only units of work which have been through restart processing. It can take the values

YES|NO

BROWSE
is an optional parameter which specified whether the inquire is for a single unit of work or for the first or next UOW in a browse, and which can take the values

FIRST|NEXT

If the BROWSE parameter is omitted, the request is a single UOW inquire. The FIRST option indicates a search for a UOW id greater than or equal to the specified UOW_ID, and next indicates a search for a UOW id greater than the specified UOW_ID.

Output parameters

RETURNED_UOW_ID
Is the unit of work for which the browse is returning status information.

UOW_STATE
indicates the state of the unit of work, and can have the values:

IN_FLIGHT|IN_DOUBT|IN_COMMIT|IN_BACKOUT

UOW_RESTART_STATE
indicates whether the unit of work has been through restart processing.

UOW_RETAINED
indicates whether the locks for the unit of work have been retained.

RESPONSE
is DFHFCDU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND CF_ACCESS_ERROR CFDT_SYSIDERR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR RESYNC_RETRY_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDU RESTART function

This function establishes recovery status for a coupling facility data table pool when a CICS region has successfully connected to it.

Input parameters

POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for recovery status is to be established. The RESTART call will be issued to the CFDT server for this pool.

POOL_NAME
is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

Output parameters

RETURNED_UOW_ID
Is the unit of work for which the browse is returning status information.

UOW_STATE
indicates the state of the unit of work, and can have the values:

IN_FLIGHT|IN_DOUBT|IN_COMMIT|IN_BACKOUT

UOW_RESTART_STATE
indicates whether the unit of work has been through restart processing.

UOW_RETAINED
indicates whether the locks for the unit of work have been retained.

RESPONSE
is DFHFCDU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED SUBSYSTEM_ALREADY_ACTIVE RESTART_ALREADY_ACTIVE TABLE_OPEN_FAILED NO_SPACE_IN_POOL CF_ACCESS_ERROR CFDT_SYSIDERR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDY RESYNC_CFDT_POOL function

This function causes a coupling facility data table pool to be resynchronized.

Input parameters

POOL_NAME is the name of the coupling facility data table pool which is to be resynchronized.

Output parameters

RESPONSE is DFHFCDY's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INITIATE_RECOVERY_FAILED TERMINATE_RECOVERY_FAILED CFDT_SERVER_CALL_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDY RESYNC_CFDT_LINK function

This function causes a link between a unit of work and a coupling facility data table pool to be resynchronized.

Input parameters

POOL_NAME is the name of the coupling facility data table pool for which the link is to be resynchronized.

UOW_ID is the unit of work ID which identifies the link to be resynchronized.

Output parameters

RESPONSE is DFHFCDY's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INITIATE_RECOVERY_FAILED TERMINATE_RECOVERY_FAILED CFDT_SERVER_CALL_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDY RETURN_CFDT_ENTRY_POINTS function

This function causes module DFHFCDY to return the entry point addresses of the other modules with which it is link-edited.

Input parameters

None

Output parameters

CFDT_EP_DFHFCDW
is the entry point address of module DFHFCDW.

CFDT_EP_DFHFCDU
is the entry point address of module DFHFCDU.

RESPONSE
is DFHFCDY's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCFL END_UOWDSN_BROWSE function

After a browse of all the data set failures within a unit of work, the END_UOWDSN_BROWSE function releases the storage that was used for a snapshot of the failures.

Input parameters

BROWSE_TOKEN
is the token which was used for the browse.

Output parameters

RESPONSE
is DFHFCFL's response to the call. It can have any of these values:

OK|INVALID|DISASTER|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_BROWSE_TOKEN
DISASTER	DISASTER_PERCOLATION ABEND

FCFL FIND_RETAINED function

This function looks for any FLAB associated with the specified data set which is flagged as retained, indicating that there are retained locks associated with the data set.

Input parameters

DSNAME
is the 44-character name of the data set for which associated retained locks are to be found.

Output parameters

RETLOCKS
indicates whether or not there are retained locks associated with the data set, and can have either of these values:

RETAINED|NORETAINED

RESPONSE
is DFHFCFL's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DISASTER_PERCOLATION ABEND

FCFL FORCE_INDOUBTS function

This function is used by the CEMT or EXEC CICS SET DSNAME() UOWACTION(COMMIT|BACKOUT|FORCE) command. Shunted in-doubt units of work are forced to complete in the specified direction. FORCE means that the direction is obtained from the ACTION specified on the transaction definition.

Input parameters

DSNAME is the 44-character name of the data set for which shunted in-doubt units of work are to be forced to complete.

DIRECTION is the direction in which the units of work are to complete: forwards (commit), backwards (backout), or heuristic (from the action specified on the transaction definition). It can have any of these values:

`FORWARD|BACKWARD|HEURISTIC`

Output parameters

RESPONSE is DFHFCFL's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DISASTER_PERCOLATION ABEND

FCFL GET_NEXT_UOWDSN function

This function returns the failure information for the next data set that has a failure within the unit of work being browsed.

Input parameters

BROWSE_TOKEN is the token for the browse, which was returned by a START_UOWDSN_BROWSE call.

Output parameters

DSNAME is the 44-character name of the data set for which failure information is returned.

[RLSACCESS] indicates whether the data set was last open in RLS or non-RLS access mode, and can have either of these values:

`RLS|NOTRLS`

[CAUSE] indicates the cause of the failure, and can have any of these values:

`CACHE|RLSSERVER|CONNECTION|DATASET|UNDEFINED`

[RETAIN_REASON]
indicates the reason for the failure, and can have any of these values:

```
RLSGONE|COMMITFAIL|IOERROR|DATASETFULL|INDEXRECFULL|
OPENERROR|DELEXITERROR|DEADLOCK|BACKUPNONBWO|
LOCKSTRUCFULL|FAILEDBKOUT|NOTAPPLIC|RR_COMMITFAIL|
RR_INDOUBT
```

RESPONSE is DFHFCFL's response to the call. It can have any of these values:

`OK|INVALID|EXCEPTION|DISASTER`

[REASON] is returned when RESPONSE is EXCEPTION, INVALID, or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	END_OF_LIST
INVALID	INVALID_BROWSE_TOKEN
DISASTER	DISASTER_PERCOLATION ABEND

FCFL RESET_BFAILS function

This function is used by the CEMT and EXEC CICS SET DSNAME() ACTION(RESETLOCKS) command. It purges shunted unit of work log records which hold backout-failure or commit-failure locks on the specified data set, and releases the locks.

Input parameters

DSNAME is the 44-character name of the data set for which backout and commit failures are to be reset.

Output parameters

RESPONSE is DFHFCFL's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DISASTER_PERCOLATION ABEND REMOVE_FAILURE

FCFL RETRY function

This function is used by the CEMT and EXEC CICS SET DSNAME() UOWACTION(RETRY) command. It drives retry of any failed backouts and commits for the specified data set, by informing DFHFCRR that the failed resource (that is, the data set) is now available.

Input parameters

DSNAME is the 44-character name of the data set for which backout and/or commits are to be retried.

Output parameters

RESPONSE is DFHFCFL's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DISASTER_PERCOLATION ABEND RESOURCE_NOT_FOUND

FCFL START_UOWDSN_BROWSE function

This function starts a browse of the data set failures within a unit of work. A snapshot of the failed data sets for the unit of work and the reasons for the failures are collected in an in-storage table to be browsed by the GET_NEXT_UOWDSN function.

Input parameters

UOW is the 8-byte local unit of work identifier.

Output parameters

BROWSE_TOKEN is a token which is used during the browse.

RESPONSE is DFHFCFL's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UOW_NOT_FOUND NO_FLABS_FOUND

RESPONSE	Possible REASON values
DISASTER	DISASTER_PERCOLATION ABEND

FCFL TEST_USER function

This function is used to test if the task has updated a record, and therefore established itself as a file user, either for any data set or for a specified data set. It can be used either as a domain subroutine call or as an inline macro.

Input parameters

[ENVIRONMENT]
is an optional parameter which is a fullword environment identifier. If specified, then the function will test whether the task is a user of any files within that environment.

[DSNAME] is an optional parameter which specifies that a particular data set is to be tested.

Output parameters

FLAB_PTR is the address of a FLAB which was found by the test. If a non-zero value is returned, then this means that the user is a task.

RESPONSE is DFHFCFL's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DISASTER_PERCOLATION ABEND

FCLJ FILE_OPEN function

This function is called when a file is opened, and causes a 'tie up record' record to be written to the log of logs if either the file (or associated data set) is forward recoverable or if autojournalling is specified for the file, to the forward recovery log if the file (or associated data set) is forward recoverable, and to the autojournal if autojournalling is specified for the file.

Input parameters

FCTE_ADDRESS is the address of the file control table entry for the file being opened.

Output parameters

RESPONSE is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR

FCLJ FILE_CLOSE Function

This function is called when a file is closed, and causes a file close log record to be written to the log of logs if either the file (or associated data set) is forward recoverable or if autojournalling is specified for the file, to the forward recovery log if the file (or associated data set) is forward recoverable, and to the autojournal if autojournalling is specified for the file.

Input parameters

FCTE_ADDRESS is the address of the file control table entry for the file being closed.

Output parameters

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR

FCLJ READ_ONLY Function

This function causes a read_only log record to be written to an autojournal, if read-only autojournalling is specified on the file definition. The log record is built using the input parameters.

Input parameters

BASE_ESDS_RBA
is the RBA of the record being read, if the file is an ESDS.

FCTE_ADDRESS
is the address of the file control table entry for the file being read.

KEY_ADDRESS
is the address of the key of the record being read.

KEY_LENGTH
is the key length of the record being read.

RECORD_ADDRESS
is the address of the record being read.

RECORD_LENGTH
is the length of the record being read.

SHUNTED
indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

YES|NO

Output parameters

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR RM_RETURNED_ERROR

FCLJ READ_UPDATE Function

This function causes a read_update log record to be written to the system log, if the file is recoverable, and if the destination parameter specifies either LOG or BOTH. It causes a read_update log record to be written to the autojournal if journaling of read updates is specified on the file definition, and if the destination parameter specifies either JOURNAL or BOTH. The log record is built using the input parameters.

Input parameters

BASE_ESDS_RBA
is the RBA of the record being read for update, if the file is an ESDS.

FCTE_ADDRESS
is the address of the file control table entry for the file being read for update.

KEY_ADDRESS
is the address of the key of the record being read for update.

KEY_LENGTH
is the key length of the record being read for update.

RECORD_ADDRESS
is the address of the record being read for update.

RECORD_LENGTH
is the length of the record being read for update.

DESTINATION specifies whether the log record is to be written to the autojournal, the system log, or both. It is used to suppress writing records that would otherwise be requested by the file definition. It can have any of these values:

JOURNAL|LOG|BOTH

SYNCHRONIZE_LOG
indicates whether or not the system log is to be synchronized (forced) when the log record is written. It can have either of these values:

YES|NO

SHUNTED indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

YES|NO

Output parameters

[LOG_TOKEN] is an optional parameter which is returned if SYNCHRONIZE(NO) was specified, and which contains a token to be used when subsequently synchronizing (forcing) the system log.

RESPONSE is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR RM_RETURNED_ERROR

FCLJ WRITE_UPDATE Function

This function causes a write_update log record to be written to the forward recovery log, if the file (or associated data set) is forward recoverable, and to the autojournal, if journaling of write updates is specified on the file definition. A write_update log record represents the completion of a file REWRITE request. The log record is built using the input parameters.

Input parameters

BACKOUT indicates if the call is made as part of transaction backout processing. It can have either of these values:

YES|NO

BASE_ESDS_RBA
is the RBA of the record being rewritten, if the file is an ESDS.

FCTE_ADDRESS is the address of the file control table entry for the file being rewritten to.

KEY_ADDRESS is the address of the key of the record being rewritten.

KEY_LENGTH is the key length of the record being rewritten to.

RECORD_ADDRESS
is the address of the record being rewritten.

RECORD_LENGTH
is the length of the record being rewritten.

SHUNTED indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

YES|NO

Output parameters

RESPONSE is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR RM_RETURNED_ERROR

FCLJ WRITE_ADD Function

This function causes a write_add log record to be written to the system log if the file is recoverable, and if the destination parameter specifies BOTH. It causes a write_add log record to be written to the autojournal if journaling of write adds was specified on the file definition. The log record is built using the input parameters.

Input parameters

BACKOUT
indicates if the call is made as part of transaction backout processing. It can have either of these values:

YES|NO

BASE_ESDS_RBA
is the RBA of the record being added, if the file is an ESDS.

FCTE_ADDRESS
is the address of the file control table entry for the file being written to.

KEY_ADDRESS
is the address of the key of the record being added.

KEY_LENGTH
is the key length of the record being written to.

MASSINSERT
indicates whether or not the record is being added as part of a mass insert. It can have either of these values:

YES|NO

DESTINATION
specifies whether the log record is to be written to the autojournal only, or to both the autojournal and the system log. It is used to suppress writing records that would otherwise be requested by the file definition. It can have either of these values:

JOURNAL|BOTH

RECORD_ADDRESS
is the address of the record being added.

RECORD_LENGTH
is the length of the record being added.

SHUNTED
indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

YES|NO

Output parameters

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR RM_RETURNED_ERROR

FCLJ WRITE_ADD_COMPLETE Function

This function causes a write_add_complete log record to be written to the forward recovery log if the file (or associated data set) is forward recoverable, and to the autojournal if write_add_complete journaling is specified on the file definition. It causes a truncated write_add_complete log record to be written to the system log if the file is a recoverable ESDS accessed in non-RLS mode. If MASSINSERT(YES) and MASSINSERT_STAGE(LAST) are specified, then only the system log record is written, and not the forward recovery log or autojournal record. The log record is built using the input parameters.

Input parameters

BACKOUT
indicates if the call is made as part of transaction backout processing. It can have either of these values:

YES|NO

BASE_ESDS_RBA
is the RBA of the record that has been added, if the file is an ESDS.

FCTE_ADDRESS
is the address of the file control table entry for the file that has been written to.

KEY_ADDRESS
is the address of the key of the record which has been added.

KEY_LENGTH
is the key length for the file which has been written to.

MASSINSERT
indicates whether or not the record was added as part of a mass insert. It can have either of these values:

YES|NO

[MASSINSERT_STAGE]
is an optional parameter which indicates whether the record is either the first or last record added during a massinsert sequence. It can have either of these values:

FIRST|LAST

RECORD_ADDRESS
is the address of the record which has been added.

RECORD_LENGTH
is the length of the record which has been added.

SHUNTED
indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

YES|NO

Output parameters

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR RM_RETURNED_ERROR

FCLJ WRITE_DELETE Function

This function causes a write_delete log record to be written to the forward recovery log if the file (or associated data set) is forward recoverable, and to the autojournal if journaling of write_deletes is specified on the file definition. The log record is built using the input parameters.

Input parameters

BACKOUT
indicates if the call is made as part of transaction backout processing. It can have either of these values:

YES|NO

BASE_ESDS_RBA
is the RBA of the record being deleted, if the file is an ESDS.

FCTE_ADDRESS
is the address of the file control table entry for the file.

KEY_ADDRESS
is the address of the key of the record being deleted.

KEY_LENGTH
is the key length for the file.

BASE_KEY_ADDRESS
is the address of the base key of the record being deleted, which is used if the data set is being accessed via a path.

SHUNTED
indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

YES|NO

Output parameters

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR RM_RETURNED_ERROR

FCLJ SYNCHRONIZE_READ_UPDATE Function

This function causes any log records previously written to the system log for this file to be synchronized (forced). The log token returned on a previous call to write a log record for this file is supplied as input.

Input parameters

FCTE_ADDRESS
is the address of the file control table entry for the file being read for update.

LOG_TOKEN
is the token returned on a previous call. The system log record written by the previous call, plus any log records written prior to that, are hardened.

Output parameters

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND RM_RETURNED_ERROR

FCLJ TAKE_KEYPOINT Function

Provided that BWO copy is supported by this CICS (indicated by a flag in file control static storage), then this function performs a scan of the file control table and, unless it has been called within the last half hour, writes a tie up record for each file open for update in non-RLS mode that is BWO-eligible and forward recoverable to the forward recovery log.

A tie up record specifies which CICS system within the sysplex opened the file, and the data set which the file was opened against. Tie up records are used by forward recovery utilities, for example CICSVR.

Input parameters

None

Output parameters

KEYPOINT_TAKEN
indicates whether or not the set of tie up records was successfully written. It can have either of these values:

YES|NO

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR TM_GETNEXT_FCTE_FAILED

FCLJ DATASET_COPY Function

This function is called when DFSMSdss initiates a copy of an RLS data set via the VSAM RLS quiesce mechanism. The function causes a 'tie up record' to be written to the log of logs if either the data set is forward recoverable, or some flavor of autojournalling has been specified in the file definition. In addition, if applicable, a record is written to the forward recovery log.

A tie up record specifies which CICS system within the sysplex opened the file, and the data set which the file was opened against. Tie up records are used by forward recovery utilities, for example CICSVR.

Input parameters

FCTE_ADDRESS
is the address of the file control table entry for the file associated with a data set being copied.

Output parameters

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR

FCQI INITIATE_QUIESCE Function

This function takes a quiesce request of type QUIESCE, IMMQUIESCE, UNQUIESCE, QUIESCE_CANCEL, NONBWO_CANCEL or BWO_CANCEL and creates a FC Quiesce Send Element (FCQSE) to describe the request. The FCQSE is added to a chain anchored in FC static, and an ECB associated with the chain (also in FC static) is posted. DFHFCQI then either suspends until the quiesce request completes or returns immediately to its caller, depending on whether busy WAIT or NOWAIT was specified on the call.

When DFHFCQI posts the ECB, the CFQS transaction (DFHFCQS) wakes up and processes the FCQSE on the chain, calling DFHFCCA QUIESCE to issue the appropriate flavor of IDAQUIES macro to SMSVSAM. When the IDAQUIES has completed, DFHFCQS will resume DFHFCQI if it was suspended, communicating the results of the IDAQUIES via the FCQSE. The FCQSE can then be unchained and freed.

Input parameters

QUIESCE_TYPE
indicates the type of quiesce being initiated and can have any of these values:

QUIESCE|IMMQUIESCE|UNQUIESCE|NONBWO_CANCEL|
BWO_CANCEL|QUIESCE_CANCEL

DSNAME
is the 44-character name of the base data set to be quiesced.

BUSY
indicates whether DFHFCQI is to wait for the quiesce to complete, or is to return immediately to the caller, and can take either of these values:

WAIT|NOWAIT

SOURCE
indicates whether the source of the quiesce request was CICS or a user, and can take either of these values:

CICS|USER

Output parameters

RESPONSE
is DFHFCQI's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID, EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_QUIESCE_TYPE
EXCEPTION	NOT_SUPPORTED UNKNOWN_VSAM_DATASET QUIESCE_NOT_POSSIBLE UNQUIESCE_NOT_POSSIBLE CANCELLED TIMED_OUT IOERR SERVER_FAILURE DATASET_MIGRATED
DISASTER	ABEND CATALOG_ERROR DISASTER_PERCOLATION

FCQI INQUIRE_QUIESCE Function

This function returns the quiesce state of a data set as QUIESCED, UNQUIESCED, or QUIESCING. DFHFCAT is called to inquire on the state of the 'quiesced' bit in the VSAM ICF catalog, which will return QUIESCED or UNQUIESCED. If UNQUIESCED is returned, the FCQSE chain is then scanned to find an FCQSE specifying the data set in question. If such an FCQSE is found for a quiesce or immquiesce request then a state of QUIESCING is returned. There is no UNQUIESCING state as the unquiesce operation is far quicker than quiesce.

Input parameters

DSNAME
is the 44-character name of the base data set for which quiesce state information is to be returned.

Output parameters

QUIESCESTATE
indicates the quiesce state of the data set, and can have any of these values:

QUIESCED|UNQUIESCED|QUIESCING

RESPONSE
is DFHFCQI's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_SUPPORTED UNKNOWN_VSAM_DATASET IOERR
DISASTER	ABEND CATALOG_ERROR DISASTER_PERCOLATION

FCQI COMPLETE_QUIESCE Function

This function is invoked whenever CICS has finished the processing for those quiesce requests for which SMSVSAM must be notified with an IDAQUIES QUICMP. Such quiesce requests are VSAM QUICLOSE (quiesce), QIOCOPY (non-BWO backup) and QUIBWO (BWO backup). This is achieved by calling DFHFCCA QUIESCE_COMPLETE to issue the IDAQUIES QUICMP macro to SMSVSAM.

Input parameters

DSNAME
is the 44-character name of the base data set for which quiesce processing has been completed by this CICS.

QUIESCE_TOKEN
is the token which was supplied by SMSVSAM when it drove the quiesce exit for the original quiesce request, and which must be returned on the IDAQUIES QUICMP.

Output parameters

RESPONSE
is DFHFCQI's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IOERR SERVER_FAILURE
DISASTER	ABEND DISASTER_PERCOLATION

FCQR RECEIVE_QUIESCES Function

This function consists of a forever loop around a dispatcher wait on an ECB. It receives work from the CICS RLS quiesce exit DFHFCQX whenever SMSVSAM requires CICS to perform processing for a quiesce request. DFHFCQX queues the request to DFHFCQR by adding an FC Quiesce Receive Element (FCQRE) to a chain anchored in file control static storage, and posting the ECB associated with the chain, also in FC static.

The posting of the ECB wakes the CFQR transaction, which executes the code in DFHFCQR. The FCQREs on the chain are processed, and DFHFCQU is called with function PROCESS_QUIESCE to perform the actual work. The ECB might also be posted to inform DFHFCQR that CICS is terminating. When DFHFCQU has finished processing, DFHFCQR unchains and frees the FCQRE.

Input parameters

None.

Output parameters

RESPONSE
is DFHFCQR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND PROCESS_QUIESCE_ERROR DISASTER_PERCOLATION

FCQS SEND_QUIESCES Function

This function consists of a forever loop around a dispatcher wait on a list of ECBs. Work is received from tasks that wish to send a quiesce request to SMSVSAM. Such tasks call DFHFCQI with function INITIATE_QUIESCE, which queues the request to DFHFCQS by adding an FC Quiesce Send Element (FCQSE) to the chain anchored in file control static storage, and posting an ECB associated with the chain, also in FC static.

When the ECB is posted, it wakes the CFQS transaction, which executes the code in DFHFCQS. The FCQSEs on the chain are processed, and DFHFCCA is called with function QUIESCE_REQUEST to issue the appropriate flavor of IDAQUIES macro to SMSVSAM. This is an asynchronous operation, and SMSVSAM returns the address of an ECB that will be posted when the IDAQUIES completes. This is saved in the FCQSE.

DFHFCQS then goes back into its dispatcher wait. It is actually waiting on a list of ECBs, the ECB for the chain plus an ECB for **each** IDAQUIES request. It wakes and processes the chain whenever one of these ECBs is posted. The wait also specifies a timeout interval, so that IDAQUIES requests that hang can be detected. When DFHFCQS wakes up, this can mean that: there is new work on the chain, or a quiesce request has completed, or a quiesce request timed out, or CICS is terminating. When a quiesce request has completed or timed out, DFHFCQS will resume the initiating task if it is waiting, after issuing appropriate messages and invoking global user exit XFCQUIS if active.

Input parameters

None.

Output parameters

RESPONSE is DFHFCQS's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND TIMEOUT_CANCEL_ERROR DISASTER_PERCOLATION

FCQU PROCESS_QUIESCE Function

DFHFCQU PROCESS_QUIESCE is called whenever a quiesce request is received from VSAM RLS. The quiesce exit DFHFCQX queues requests to the CFQR system transaction (DFHFCQR), which calls DFHFCQU to process each one in turn. The PROCESS_QUIESCE function is also called to implement a non-RLS variant of QUIESCE called NON_RLS_CLOSE. This is for non-RLS files, is only used internally by CICS, and does not run under the CFQR system transaction. Each quiesce request type is processed in a different way by DFHFCQU.

QUIESCE corresponds to an SMSVSAM QUICLOSE. All files open against the data set are closed, the file state of each file is set to unenabled but with a flag that says re-enable on QUIOPEN, and a QUICMP is issued for the QUICLOSE back to VSAM RLS to indicate our QUICLOSE processing is complete. The immediate option on the DFHFCQU call governs how file closes are to be performed. If NO or omitted then closes will occur when all UOWs using the data set have completed normally. If YES then all such UOWs will be force purged to speed things up.

UNQUIESCE corresponds to an SMSVSAM QUIOPEN. All files associated with the data set are checked to see if the file state requires resetting back to enabled, because it had been set unenabled by a QUICLOSE.

NONBWO_START corresponds to an SMSVSAM QUICOPY. CICS prepares for a non-BWO backup of the data set by preventing new units of work from updating the data set, allowing existing UOWs to finish updating the data set, and then issuing a QUICMP for the QUICOPY back to SMSVSAM to indicate that QUICOPY processing is complete. The files involved are not closed.

NONBWO_END corresponds to an SMSVSAM QUICEND. All files associated with the data set are checked to see if the file state requires resetting to enabled because it had been set unenabled by an OPEN failure, and a set of ’tie up records’ are written for the data set.

BWO corresponds to an SMSVSAM QUIBWO. CICS prepares for a BWO backup of the data set by writing a set of ’tie up records’ allowing existing units of work to finish updating the data set, and then issuing a QUICMP for the QUIBWO back to SMSVSAM to indicate that QUIBWO processing is complete. The files involved are not closed, nor are updates prevented.

BWO_END corresponds to an SMSVSAM QUIBEND. The only processing involved is to stop an existing BWO quiesce if one is in progress.

LOST_LOCKS_RECOVERED
corresponds to an SMSVSAM QUILLRC. It notifies CICS that lost locks recovery has been completed for the data set throughout the sysplex. DFHFCRR is called with function LOST_LOCKS_RECOVERED to process the availability of the data set.

FORWARD_RECOVERY_COMPLETE
corresponds to an SMSVSAM QUIFRC. It notifies CICS that forward recovery has been completed for the data set. DFHFCRR is called with function RESOURCE_AVAILABLE to process the availability of the data set.

CACHE_AVAILABLE
corresponds to an SMSVSAM QUICA. It notifies CICS that a previously failed cache structure is now available. DFHFCRR is called with function RESOURCE_AVAILABLE to process the availability of the cache.

NON_RLS_CLOSE
processes a non-RLS variant of type CLOSE called NON_RLS_CLOSE. All ACBs open against the specified non-RLS data set are closed.

Some of the requests cause global user exit XFCVSDS to be invoked if active and a DSNB exists for the data set, and XFCVSDS can suppress certain of the requests if desired. Suppression causes the quiesce request to be cancelled throughout the sysplex (by issuing the inverse quiesce request).

The types of quiesce that DFHFCQU can receive fall into two 'completion' categories.

1. Those for which VSAM does not require completion notification. For these no IDAQUIES QUICMP is issued. The successful return of the quiesce exit DFHFCQX to VSAM is sufficient. The requests in this category are:

   UNQUIESCE, NONBWO_END, BWO_END, CACHE_AVAILABLE,
   LOCKS_RECOVERY_COMPLETE, FORWARD_RECOVERY_COMPLETE.

2. Those for which VSAM requires completion notification because CICS must complete some critical processing. For these an IDAQUIES QUICMP must be issued when CICS processing is complete. The requests in this category are:

   QUIESCE, NONBWO_START, BWO_START.

Input parameters

QUIESCE_TYPE
indicates the type of quiesce being requested. It can have any of these values:

QUIESCE|UNQUIESCE|NONBWO_START|NONBWO_END|BWO_START|
BWO_END|LOCKS_RECOVERY_COMPLETE|
FORWARD_RECOVERY_COMPLETE|CACHE_AVAILABLE|
NON_RLS_CLOSE

DSNAME|CACHE_NAME
either specifies the 44-character name of the data set to which the quiesce request applies, or (when the quiesce_type is CACHE_AVAILABLE) the 16-character name of the cache structure which has become available.

[IMMEDIATE]
applies when the quiesce_type is QUIESCE or NON_RLS_CLOSE, and indicates whether units of work which have updated the data set will be forced to complete immediately, or whether the request will wait for such units of work to complete naturally. It can have either of these values:

YES|NO

[CONCURRENT]
applies when the quiesce_type is NONBWO_START or BWO_START, and indicates whether the concurrent copy technique is being used. It is purely informational, and has no effect on the processing. It can have either of these values:

YES|NO

[QUIESCE_TOKEN]
is a token which is supplied by SMSVSAM when certain quiesce requests are initiated, and must be passed back when the quiesce complete is issued.

Output parameters

RESPONSE
is DFHFCQU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID, EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_QUIESCE_TYPE
EXCEPTION	DSNB_NOT_FOUND
DISASTER	ABEND DISASTER_PERCOLATION DFHFCRR_ERROR DFHFCQI_ERROR DFHFCFS_ERROR DFHTM_FAILURE

FCRR RESTART_RLS Function

This function performs a restart of the RLS component of file control. The exact processing depends on the type of restart being performed.

COLD and INITIAL

The RLS control ACB is registered, and RLS is cold started, both via calls to DFHFCCA.

WARM and EMERGENCY

The RLS control ACB is registered, and recovery information is inquired upon from SMSVSAM, both via calls to DFHFCCA. If the recovery information indicates that there are data sets in lost locks status, then the corresponding DSNBs are marked as being lost locks, and preparation for lost locks recovery is carried out. Any orphan locks are eliminated.

DYNAMIC

This type of restart occurs when a new instance of the SMSVSAM server becomes available following a previous server failure.

Having waited for file control restart to complete if it was still in progress, and for any in-progress dynamic RLS restart to complete, RLS access is drained if this has not already been done, the control ACB is registered, and recovery information is inquired upon from SMSVSAM, all three via calls to DFHFCCA. If the recovery information indicates that there are data sets in lost locks status, then the corresponding DSNBs are marked as being lost locks, and preparation for lost locks recovery is carried out. Any orphan locks are eliminated. The CICS recovery manager is called to unshunt any units of work that are backout-failed due to the SMSVSAM server failure or a general file backout failure, and any units of work that are commit-failed due to the SMSVSAM server failure.

Input parameters

TYPE_OF_RESTART
: indicates the type of RLS restart being performed, and can have any of these values:

`COLD|WARM|EMERGENCY|DYNAMIC`

Output parameters

RESPONSE
: is DFHFCRR's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
: is returned when RESPONSE is INVALID, EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION INVALID_RESTART_TYPE

RESPONSE	Possible REASON values
EXCEPTION	REGISTER_CTL_ACB_FAILED COLD_START_RLS_FAILED DRAIN_RLS_FAILED LOST_LOCKS_INFO_LOST INQUIRE_RECOVERY_FAILED LOST_LOCKS_COMPLETE_FAILED ORPHAN_RELEASE_FAILED
DISASTER	DSSR_FAILED TM_LOCATE_FAILED TM_UNLOCK_FAILED ABEND DISASTER_PERCOLATION

FCRR RESOURCE_AVAILABLE function

This function causes the CICS recovery manager to be notified of the availability of the specified resource. When the resource_type is DSET, an RMRE AVAIL call is issued for the specified data set. When the resource_type is CACHE, an RMRE avail call is issued for every data set that has outstanding work shunted due either to a cache failure or to a general file backout failure. When the resource_type is OTHER, an RMRE AVAIL call is issued for the specified resource.

Input parameters

RESOURCE_TYPE
is the type of resource which has become available, and can have any of these values:
DSET|CACHE|OTHER

RESOURCE_NAME
is the 44-character field containing the name of the resource which has become available.

RESOURCE_NAME_LENGTH
is a halfword containing the actual length of the resource name.

Output parameters

RESPONSE
is DFHFCRR's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION INVALID_RESOURCE_TYPE
DISASTER	ABEND DISASTER_PERCOLATION

FCRR LOST_LOCKS_RECOVERED function

This function is called when lost locks recovery for a data set has been completed by all CICS regions that were sharing it, and causes the flag in the DSNB which indicates that the data set is in lost locks state to be cleared.

Input parameters

RESOURCE_NAME
is the 44-character field containing the name of the resource (data set) for which lost locks recovery has been completed.

Output parameters

RESPONSE
is DFHFCRR's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID, EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION
EXCEPTION	SPHERE_UNKNOWN
DISASTER	TM_LOCATE_FAILED TM_UNLOCK_FAILED ABEND DISASTER_PERCOLATION

File Control's call back gates

Table 8 summarizes file control's call back gates. It shows the FC level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the format for calls to the gate.

Table 8. File control's call back gates

Gate	Trace	Function	Format
RMRO	FC 0BE0 FC 0BE1	PERFORM_PREPARE PERFORM_COMMIT START_BACKOUT DELIVER_BACKOUT_DATA END_BACKOUT PERFORM_SHUNT PERFORM_UNSHUNT	RMRO
RMKP	FC 0BE0 FC 0BE1	TAKE_KEYPOINT	RMKP
RMLK	FC 24A0 FC 24A1	PREPARE COMMIT SEND_DO_COMMIT SHUNT UNSHUNT	RMLK
RMDE	FC 0BE0 FC 0BE1	START_DELIVERY DELIVER_RECOVERY DELIVER_FORGET END_DELIVERY	RMDE
LGGL	FC 2350 FC 2351	ERROR	LGGL
DMEN	FC 0BD0 FC 0BD1	NOTIFY_SMSVSAM_AVAILABLE	DMEN

You can find descriptions of these functions and their input and output parameters, in the chapters on the recovery manager, log manager, and domain manager.

The functions of the RMRO gate are processed by DFHFCRC. For PERFORM_PREPARE and PERFORM_COMMIT, DFHFCRC performs prepare and commit processing respectively for any file resources involved in the unit of work. For START_BACKOUT, DELIVER_BACKOUT_DATA and END_BACKOUT, DFHFCRC backs out changes made to file resources by the unit of work. For PERFORM_SHUNT and PERFORM_UNSHUNT, DFHFCRC respectively shunts and unshunts the file control structures representing recoverable parts of the unit of work.

The functions of the RMKP gate are processed by DFHFCRC. For TAKE_KEYPOINT, DFHFCRC performs processing required for forward recovery of BWO-eligible non-RLS files.

The functions of the RMLK gate are processed by DFHFCDW, which performs syncpoint and recovery functions for recoverable coupling facility data tables.

The functions of the RMDE gate are passed through by DFHFCRC to DFHFCIR. For START_DELIVERY, DFHFCIR takes no action. For DELIVER_RECOVERY and DELIVER_FORGET, DFHFCIR uses the log records that are delivered to it to rebuild file control structures representing the recoverable parts of each unit of work, and also rebuilds locks for non-RLS files. For END_DELIVERY, DFHFCIR notifies file control that the rebuilding of recovery information at CICS restart is now complete.

The functions of the LGGL gate are processed by DFHFCLF. For ERROR, DFHFCLF takes actions to handle a log stream failure for a general log used by file control.

The functions of the DMEN gate are processed by DFHFCES. For NOTIFY_SMSVSAM_AVAILABLE, DFHFCES calls DFHFCRR with a function of RESTART_RLS and TYPE_OF_RESTART as DYNAMIC.

Exits

The following global user exit points are provided for file control:

In DFHEIFC XFCREQ and XFCREQC
In DFHFCFS XFCSREQ and XFCSREQC
In DFHFCN XFCNREC
In DFHFCRC XFCBFAIL, XFCBOUT, XFCBOVER and XFCLDEL

The following global user exit points are provided specifically for data table services: XDTAD, XDTLC, and XDTRD.

See the *CICS Customization Guide* for further information.

Trace

The following point IDs are provided for file control:

- AP 04xx, for which the trace levels are FC 1, FC 2, and Exc
- AP 0Bxx, for which the trace levels are FC 1, FC 2, and Exc.
- AP 23xx, for which the trace levels are FC 1, FC 2, and Exc.
- AP 24xx, for which the trace levels are FC 1, FC 2, and Exc.

Note: Trace entries for shared data table services have point IDs at the lower end of the AP 0Bxx range, and a corresponding trace level of FC 2. Trace entries for coupling facility data tables are from AP 2440 upwards.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 25. Front end programming interface (FEPI)

The front end programming interface (FEPI) is an integral part of CICS Transaction Server for z/OS, Version 3 Release 1. The function is called a front end programming interface because it enables you to write CICS application programs that access other CICS or IMS programs. In other words, it provides a front end to those programs.

Design overview

This section describes how FEPI works at a high level. It discusses how the FEPI functions are provided within CICS.

FEPI as a CICS transaction

The main functions of FEPI are provided through the **CSZI** transaction, which is defined in group DFHFEPI. CSZI runs the FEPI Resource Manager, which is responsible for most of the functions of FEPI.

The FEPI Resource Manager transaction is attached during a late stage of CICS initialization. CSZI runs as a high-priority CICS system task, and cannot be canceled by an operator; it is terminated during CICS shutdown processing.

The FEPI commands communicate with the Resource Manager through the FEPI adapter program, which is loaded when CICS initializes, and is part of the CICS nucleus.

The FEPI adapter receives information from FEPI commands through two EXEC stubs, **DFHESZ** and **DFHEIQSZ**. DFHESZ handles the FEPI application programming commands, while DFHEIQSZ handles the system programming commands.

These two EXEC stubs call the adapter to do FEPI work. The adapter communicates with the Resource Manager through work queues. See “Application flows” for details of these flows.

Application flows

“FEPI as a CICS transaction” outlined the main components of FEPI. This section shows the pathways followed by a FEPI command.

Application programming command flows

The FEPI application programming commands flow through the normal EXEC CICS route into DFHEIP, from where they are routed to DFHESZ. DFHESZ passes the command parameter list to the FEPI adapter. After checking and other processing, the adapter generates another parameter list in internal format, and places it on a queue for the FEPI Resource Manager to process.

While the adapter is waiting for the Resource Manager to process the command, it issues a wait. The event control block (ECB) for this wait is contained in the parameter list queued to the Resource Manager. Consequently, the application that issued the FEPI command is in a wait state while the Resource Manager is processing the FEPI command. For information about wait processing, see the *CICS Problem Determination Guide*.

When the Resource Manager has retrieved the command from its queue, and processed it, the ECB is posted, thus ending the wait.

Control returns from the adapter to DFHEIP, and the application program in the normal fashion.

Figure 51 on page 264 shows this processing. Note that the details are for illustration only.

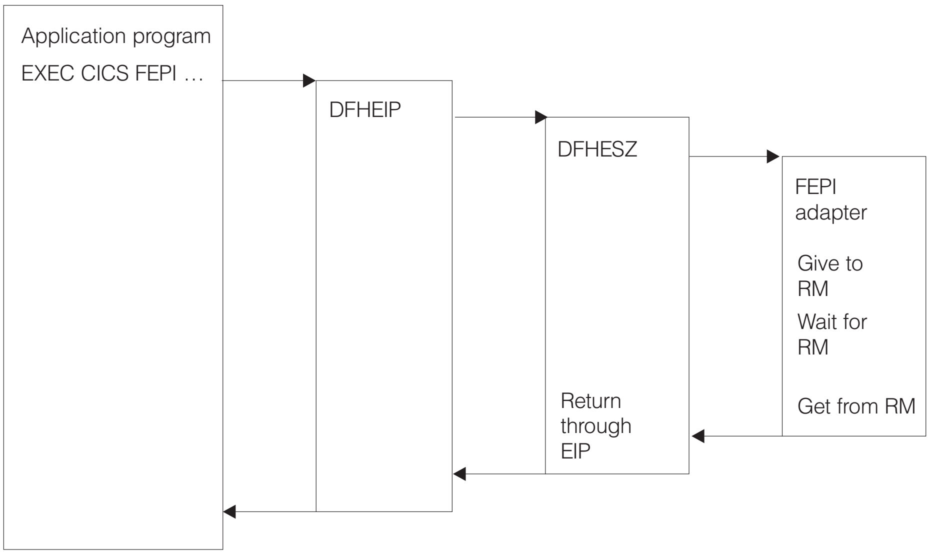


Figure 51. FEPI application programming command flows

System programming command flows

The FEPI system programming commands flow through DFHEIQSZ rather than DFHESZ, but the overall picture is the same as for FEPI application programming requests.

However, some system commands can flow directly to the FEPI Resource Manager, bypassing the EXEC stub. These commands are mainly concerned with FEPI processing to be done at “special” events, such as task termination and CICS shutdown.

Figure 52 shows this processing. The details are for illustration only.

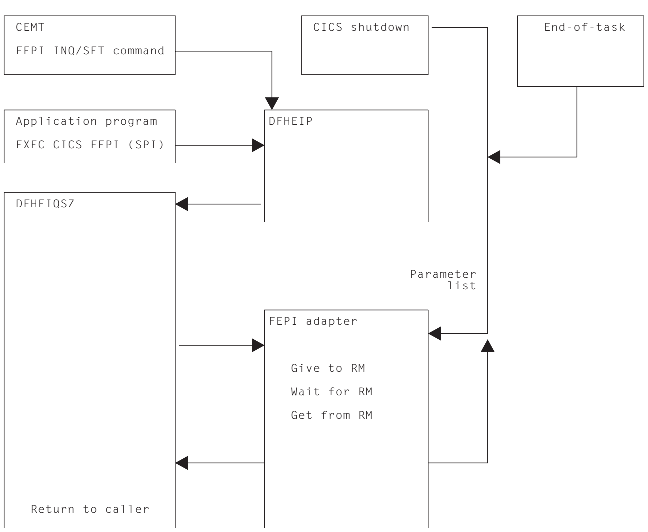


Figure 52. FEPI system programming command flows

Logic flow within the FEPI adapter

Figure 53 shows the logic flow within the FEPI adapter in more detail. In particular, it shows the points at which the FEPI global user exits, XSZBRQ and XSZARQ, and the FEPI journaling function, are invoked.

Journaling of data occurs after the Resource Manager has processed the request, but before XSZARQ is called (if active). Data is not journaled if your XSZBRQ exit program rejects the request.

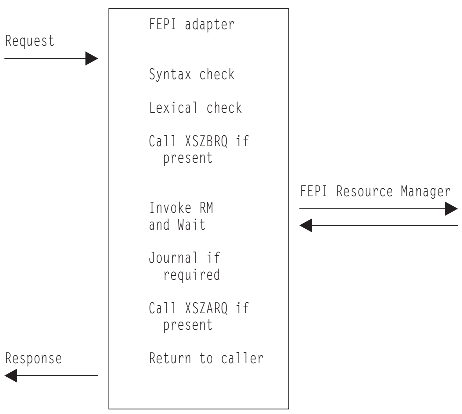


Figure 53. Logic flow within the FEPI adapter

The FEPI adapter and Resource Manager

The FEPI adapter runs as part of the invoking CICS task, and so runs under the **QR** task control block (TCB). The FEPI Resource Manager, running as CSZI, runs under the **SZ** TCB (reserved for use by the Resource Manager).

Consequently, the interface between the adapter and the Resource Manager uses waits and queues to synchronize access. The control block used to pass information between the adapter and the Resource Manager is called the **DQE**.

Figure 54 shows this interaction. The details are for illustration only.

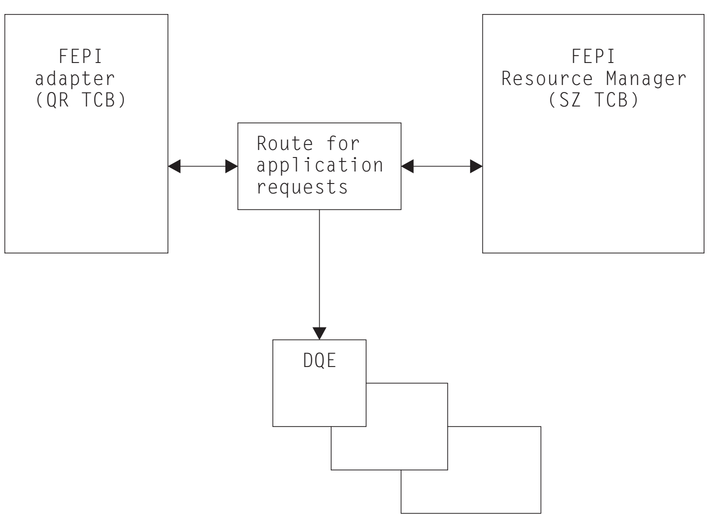


Figure 54. Interaction of the FEPI adapter and Resource Manager

The FEPI Resource Manager work queues

When organizing its work, the FEPI Resource Manager uses a mechanism that is optimized for the FEPI environment. Each DQE is chained to a queue representing the work to be done next.

The most common mechanism used for this movement between queues is the connection on which the original FEPI command is operating.

Summary of Resource Manager work queues

In addition to the application queue, there are other queues used only by the Resource Manager. They are:

API/Norm	Used for FEPI application requests
API/Expd	Used for FEPI high-priority application requests
PRB	Used for Resource Manager internal work
PRB/Time	Used for Resource Manager internal time-dependent work
IRB	Used to control work done in VTAM exits
IRB/Time	Used to control time-dependent work done in VTAM exits
TPEND8	Used to process VTAM TPEND8 conditions
Timer	Used to control timer-related work
Free	Used to hold VTAM RBs that have to be freed
Discard	Used to control requests initiated by FEPI DISCARD commands.
CICS work	Used to schedule work that has to run under the CICS QR TCB.

Control blocks

This section lists *some* of the FEPI control blocks and their resident storage subpools, where applicable. For details of the subpools, see Chapter 104, “Storage manager domain (SM),” on page 1143.

DFHSZSDS (Static area)	Used to anchor all FEPI storage
DFHSZDCM (Common area)	Used to anchor all FEPI Resource Manager storage (*SZSPFCCM*)
DFHSZDND (Node)	Represents a node (*SZSPFCND*)
DFHSZDPD (Pool)	Represents a pool (*SZSPFCPD*)
DFHSZDTD (Target)	Represents a target (*SZSPFCTD*)
DFHSZDPS (Propertyset)	Represents a property set (*SZSPFCPS*)
DFHSZDCD (Connection)	Represents a connection (a node-target pair) (*SZSPFCCD*)
DFHSZDCV (Conversation)	Represents a FEPI conversation (*SZSPFCCV*)
DFHSZDSR (Surrogate)	Used to associate nodes, pools, and targets with other control blocks—*not* to be confused with a CICS surrogate terminal (*SZSPFCSR*)
DFHSZDQE (Queue element)	Used to schedule Resource Manager work (*SZSPFCWE*).

Some of the relations between FEPI control blocks are shown in Figure 55 on page 267.

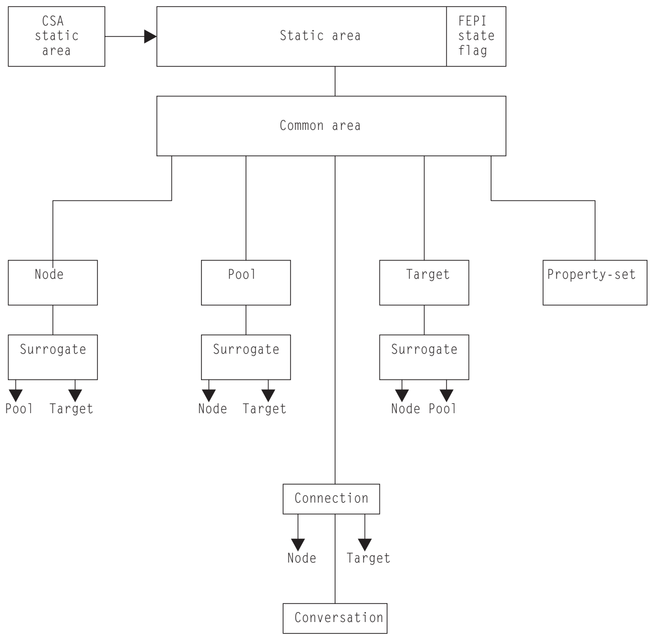


Figure 55. FEPI control block relationships

Dump

This section documents the areas that can be listed by the FEPI dump routines. For information about how to use these facilities for problem determination, see the *CICS Problem Determination Guide*.

Note: The length of areas described in this section may change in future versions or releases of CICS. Any status values interpreted may also be open to change. So you should use diagrams and descriptions in this section only as *illustrations* of how to interpret FEPI dumps.

Here is a list all the FEPI areas that can be interpreted. If an area does not exist in your system, it does not appear in the dump—no error message is produced.

- The static area
- The common area:
  - The temporary ACB.
- Property sets
- Pools:
  - Connections within the pool
  - Node surrogates chained to the pool
  - Target surrogates chained to the pool
  - Queued allocate DQEs waiting within the pool
- Nodes:
  - Connections used by the node
  - Pool surrogates chained to the node
  - Node's ACB
  - Node's RPL
  - Unsolicited BINDs queued to the node
- Targets:
  - Connections used by the target
  - Connections queueing on the target
  - Pool surrogates chained to the target

- Connections:
  - Current API request
  - Connection's RPL
  - Connection's RESP data
  - Formatted data extension:
    - Graphics plane
    - Attributes
    - Highlights
    - Color
    - Selection
    - Validation
- Active conversations
- Browse conversations
- Inactive conversations
- CICS work queues
- PRB DQEs
- PRB time DQEs
- IRB DQEs
- IRB time DQEs
- TPend8 DQEs
- Discard DQEs
- API normal DQEs
- API expd DQEs
- Timer DQEs
- Free RBs
- The stacks (level 2 only).

A DQE is interpreted further, as follows:
- The DRP representing the DQE
- The DQE associated storage
- Any horizontal DQE extension (chained) DQEs.

The following sections describe *some* of the areas interpreted.

The static area

```
==SZ.Static FEPI Static Area
 SZSDS 03AF6710 FEPI Static Area (Status is Open) 1
    0000  01406EC4 C6C8E2E9 E2C4E240 40404040  00000003 00030001 00000000 00000000  *. >DFHSZSDS     ................*    03AF6710
    0020  00000000 03B78000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03AF6730
    0040  04B96440 000000DC 04B964F0 000000DD  04B965A0 000000DE 04B96650 000000DF  *... .......0..............&....*    03AF6750
    0060  04B96700 000000E0 04B967B0 000000E1  04B96860 000000E2 04B96910 000000E3  *..................-...S.......T*    03AF6770
    0080  04B969C0 000000E4 04B96A70 000000E5  04B96B20 000000E6 04B96BD0 000000E7  *.......U.......V..,....W..,....X*    03AF6790
    00A0  04B96C80 000000E8 04B96DE0 000000EA  04B96E90 000000EB 04B96D30 000000E9  *..%....Y.._.......>......_....Z*    03AF67B0
    00C0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03AF67D0
    00E0 -    013F LINES SAME AS ABOVE                                                                                        03AF67F0
```

1 This shows the status of the FEPI system; that is, whether or not it was running.

The common area

```
==SZ.Common FEPI Common Area
 SZDCM 1AE1A000 FEPI Common Area
    0000  01A86EC4 C6C8E2E9 C4C3D400 00000000  1C0E55A0 000000DB 1AE1A000 00000000  *.y>DFHSZDCM.....................*    1AE1A000
    0020  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    1AE1A020
    0040  00000000 00000000 00000000 1AE1D000  1BEE7210 1BEE72F0 1BEE7280 1BEE7360  *.......................0.......-*    1AE1A040
    0060  1AE1B000 00000000 00000000 1AE345A0  00000000 00000000 00000000 1AE2F000  *.............T...............S0.*    1AE1A060
    0080  00000000 1AD7F710 1BEE71C8 0007A630  00010000 0000000E 00001000 00000000  *.....P7....H..w.................*    1AE1A080
    00A0  00000000 00000F70 0000000E 00000000  1AC9F990 1AE18000 0000006C 1BEEB1B0  *.................I9........%....*    1AE1A0A0
    00C0  1AE1A000 1AE182B0 00000028 1BEEB20F  1AE2CB5C 1BEE71C8 0000000B 00000000  *......b..........S.*...H........*    1AE1A0C0
    00E0  9BEEA56E 1AE2CA28 1AE18000 1AE2CA28  9BEEAF0E 00000000 1AE1A114 1AE1A118  *..v>.S.......S..................*    1AE1A0E0
    0100  1AE1A11C 1AE1A120 1AE1A124 1AE1A128  9AE1A12C 00000000 00000000 00000000  *................................*    1AE1A100
    0120  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    1AE1A120
    0140  00000000 00000000 00000000 00000000  13000001 1AE1A16C 1AE1A168 1BF023C8  *.......................%.....0.H*    1AE1A140
    0160  1AE1A000 00000000 00000064 0002F6C1  00000000 00000000 00000000 00000000  *..............6A................*    1AE1A160
    0180  00002000 00010000 00000000 00000001  00000000 0000003C 00000078 00000005  *................................*    1AE1A180
    01A0  00000000 00000000                                                         *........                        *    1AE1A1A0
Dispatcher Status is Running,CICS Trigger(No ),Recovery Trigger(No ). Receive-Any Size = 00004096 1
Current Request is at address 1AE2F000 Current Timer Element is at address 00000000 2
Exit footprints :  TPEND, NSEXIT, SCIP, LOSTTERM, RECVANY, Common, DFASY, SETLOGON 3
LU2  footprints :  Send, Drain, REC(Spec), REQSESS, OPNSEC
LUP  footprints :  Send, Drain, REC(Spec), REQSESS, OPNSEC
RPL  footprints :  REQSESS, RA(A) issue, UnSolBind, RA(A) fdbk, IRB fdbk
```

1 This shows whether the FEPI Resource Manager was active.

2 This shows details of the currently executing item.

3 This shows whether one of the VTAM exits was active (none in this case). If an exit was active, it is shown preceded by an equals (=) character.

Property sets

```
==SZ.Prop FEPI Propertysets
 SZDPS.YRAH1 03B98370 FEPI Propertyset
    0000  00706EC4 C6C8E2E9 C4D7E200 00000000  04B96A70 000000E5 03B98370 00000000  *..>DFHSZDPS............V..c.....*    03B98370
    0020  03B98420 03B982C0 E8D9C1C8 F1404040  00000000 00000000 00000000 00000000  *..d...b.YRAH1   ................*    03B98390
    0040  00000000 00001000 00000004 0215021E  02200222 02120214 00000000 00000000  *................................*    03B983B0
    0060  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03B983D0
    0080  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03B983F0
    00A0  00000000 00000000 00000000 00000000                                       *................                *    03B98410
```

This shows details of the property set, which defines the characteristics of FEPI pools.

Pools

```
==SZ.Pool FEPI Pools
 SZDPD.P1 03B97000 FEPI Pool (created from Propertyset Y1      ) 1
    0000  008C6EC4 C6C8E2E9 C4D7C400 00000000  04B969C0 000000E4 03B97000 00000000  *..>DFHSZDPD............U........*    03B97000
    0020  03B97110 00000000 D7F14040 40404040  E8F14040 40404040 03BA9440 03BA9000  *........P1      Y1      ..m ....*    03B97020
    0040  03BAA2E0 00000000 00490226 00000000  00000000 00000000 00000000 00000000  *..s.............................*    03B97040
    0060  00001000 00000003 0215021E 02200222  02120214 00000000 00000000 00000000  *................................*    03B97060
    0080  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03B97080
    00A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03B970A0
    00C0  00000000 00000000 00000000 F1F2F3F4  F5F6F7F8 F9F0F1F2 F3F4F5F6 F7F8F9F0  *............12345678901234567890*    03B970C0
    00E0  F1F2F3F4 F5F6F7F8 F9F0F1F2 F3F4F5F6  F7F8F9F0 F1F2F3F4 F5F6F7F8 F9F0F1F2  *12345678901234567890123456789012*    03B970E0
    0100  F3F4F5F6 F7F8F9F0 F1F2F3E7                                                *34567890123X                    *    03B97100
          User Data is :12345678901234567890123456789012345678901234567890123X
          Pool is using the Connection at address 03BAA2E0 2
 SZDSR 03BA9440 FEPI Pool's Surrogate refers to Node IYAEZM42 at address 03B9C1C0 3
    0000  003C6EC4 C6C8E2E9 C4E2D900 00000000  04B96C80 000000E8 03BA9440 00000000  *..>DFHSZDSR.......%....Y..m ....*    03BA9440
    0020  00000000 03BA9400 03BA95C0 00000000  03B97000 03B9C1C0 00000000           *......m...n...........A.....    *    03BA9460
 SZDSR 03BA9000 FEPI Pool's Surrogate refers to Target CSYSE6   at address 03BA8000 4
    0000  003C6EC4 C6C8E2E9 C4E2D900 00000000  04B96C80 000000E8 03BA9000 00000000  *..>DFHSZDSR.......%....Y........*    03BA9000
    0020  00000000 00000000 03BA9040 00000000  03B97000 03BA8000 00000000           *........... ................    *    03BA9020
5
```

1 Refers to the property set that defines the characteristics of the pool.

2 Shows the connections within the pool.

3 Shows the node surrogates chained to the pool.

4 Shows the pool surrogates chained to the pool.

5 There are no ACB, RPL, and unsolicited bind areas for this pool (no queued allocated DQEs within the pool). If there were, the areas would be shown here.

Nodes

```
==SZ.Node FEPI Nodes
 SZDND.IYAEZM42 03B9C1C0 FEPI Node
    0000  00946EC4 C6C8E2E9 C4D5C400 00000000  04B96910 000000E3 03B9C1C0 00000000  *.m>DFHSZDND...........T..A.....*    03B9C1C0
    0020  00000000 00000000 00000068 00000000  00000000 00000000 00000000 00000000  *................................*    03B9C1E0
    0040  00020080 03BB4F00 03B9C2A0 03B9C0E0  03B9FCF0 03BA9C40 000641A0 03B80000  *......|...B........0... ........*    03B9C200
    0060  00064160 08C9E8C1 C5E9D4F4 F2000000  00404040 40404040 40000000 00490045  *...-.IYAEZM42....        .......*    03B9C220
    0080  00450226 00450000 00000000 00000001  00000000 F1F2F3F4 F5F6F7F8 F9F0F1F2  *...................123456789012*    03B9C240
    00A0  F3F4F5F6 F7F8F9F0 F1F2F3F4 F5F6F7F8  F9F0F1F2 F3F4F5F6 F7F8F9F0 F1F2F3F4  *34567890123456789012345678901234*    03B9C260
    00C0  F5F6F7F8 F9F0F1F2 F3F4F5F6 F7F8F9F0  F1F2F3E7                            *5678901234567890123X            *    03B9C280
          User Data is :1234567890123456789012345678901234567890123456789012345678901234567890123X
          Node is using the Connection at address 03B9FCF0 1
 SZDSR 03BA9440 FEPI Node's Surrogate refers to Pool P1       at address 03B97000 2
    0000  003C6EC4 C6C8E2E9 C4E2D900 00000000  04B96C80 000000E8 03BA9440 00000000  *..>DFHSZDSR.......%....Y..m ....*    03BA9440
    0020  00000000 03BA9400 03BA95C0 00000000  03B97000 03B9C1C0 00000000           *......m...n...........A.....    *    03BA9460
 SZDAC.IYAEZM42 00064160 FEPI Node's ACB 3
    0000  00AC6EC4 C6C8E2E9 C4C1C300 00000000  04B96440 000000DC 00064160 00000000  *..>DFHSZDAC........ .......-....*    00064160
    0020  00000000 00000000 08C9E8C1 C5E9D4F4  F2000000 00404040 40404040 40000000  *.........IYAEZM42....        ...*    00064180
    0040  A020006C 00000000 80D42000 94000001  00000000 00000000 80000008 00000000  *...%.....M..m..................*    000641A0
    0060  00000000 044B7CC0 FF000060 00000057  12000000 00000000 00000000 00000000  *......@....-....................*    000641C0
    0080  00000000 00000000 00064188 41F00020  07FE0000 00000000 00000000 00BC4FB0  *..........h.0................|.*    000641E0
    00A0  7F714F30 03B9C1C0 00000000                                               *".|...A.....                    *    00064200
 SZDRA.IYAEZM42 03BB4F00 FEPI Node's RPL 4
    0000  10006EC4 C6C8E2E9 D9C14000 00000000  04B96700 000000E0 03BB4F00 00000000  *..>DFHSZRA ...............|.....*    03BB4F00
    0020  00000000 00000000 00000000 00000000  00000000 00000000 03B80000 03B9FCF0  *..............................0*    03BB4F20
    0040  03B9C1C0 00000000 00202270 844C9E82  844B4370 00000000 00801004 04008000  *..A.........d<.bd..............*    03BB4F40
    0060  000641A0 4B800000 03BB4FB8 050001F4  28800000 00000000 00000000 00000F48  *..........|....4................*    03BB4F60
    0080  12309450 00000000 80800001 20000000  00000000 00000000 00000000 00000000  *..m&;...........................*    03BB4F80
    00A0  00000000 03B9FCF0 80004043 08130000  00000000 00000000                    *.......0.. .............        *    03BB4FA0
5
```

1 Shows the connections used by the node.

2 Shows the pool surrogates chained to the node.

3 Shows the node's ACB.

4 Shows the node's RPL.

5 There are no unsolicited binds outstanding for this nodes. If there were, the areas would be shown here.

Targets

```
==SZ.Target FEPI Targets
 SZDTD.CSYSE6 03BA8000 FEPI Target (network id is CSYSE6  ) 1
    0000  00786EC4 C6C8E2E9 C4E3C400 00000000  04B96D30 000000E9 03BA8000 00000000  *..>DFHSZDTD......._....Z........*    03BA8000
    0020  00000000 00000000 0000006C 00000000  00000000 00000000 00000000 00000000  *...........%....................*    03BA8020
    0040  00000000 00000001 03BA80C0 00000000  03BA92C0 03BA22E0 C3E2E8E2 C5F64040  *.................k.....CSYSE6  *    03BA8040
    0060  C3E2E8E2 C5F64040 00490226 00000000  00000000 00000000 F1F2F3F4 F5F6F7F8  *CSYSE6  ................12345678*    03BA8060
    0080  F9F0F1F2 F3F4F5F6 F7F8F9F0 F1F2F3F4  F5F6F7F8 F9F0F1F2 F3F4F5F6 F7F8F9F0  *90123456789012345678901234567890*    03BA8080
    00A0  F1F2F3F4 F5F6F7F8 F9F0F1F2 F3F4F5F6  F7F8F9F0 F1F2F3E7                   *1234567890123456789012 3X       *    03BA80A0
          User Data is :1234567890123456789012345678901234567890123456789012345678901234567890123X
          Target is using the Connection at address 03BA22E0 2
3
 SZDSR 03BA9000 FEPI Target's Surrogate refers to Pool P1       at address 03B97000 4
    0000  003C6EC4 C6C8E2E9 C4E2D900 00000000  04B96C80 000000E8 03BA9000 00000000  *..>DFHSZDSR.......%....Y........*    03BA9000
    0020  00000000 00000000 03BA9040 00000000  03B97000 03BA8000 00000000           *........... ................    *    03BA9020
```

1 Shows the applid (network id) of the target.

2 Shows the connections used by the target.

3 There are no connections queuing on the target. If there were, the areas would be shown here.

4 Shows the pool surrogates chained to the target.

Connections

```
==SZ.Conn FEPI Connections
 SZDCD 03BAA2E0 FEPI Connection
    0000  01286EC4 C6C8E2E9 C4C3C400 00000000  04B964F0 000000DD 03BAA2E0 00000000  *..>DFHSZDCD........0......s.....*    03BAA2E0
    0020  00000000 00000000 00000064 00000000  00000000 00000000 00000000 00000001  *................................*    03BAA300
    0040  00000000 00000000 000000C0 C0000810  00000000 03B9F000 00000000 00000000  *.....................0..........*    03BAA320
    0060  00000000 00000000 050001EB 00000000  00000000 00010000 00000000 12950000  *.............................n..*    03BAA340
    0080  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03BAA360
    00A0  00000000 00000000 00000000 C4F4C1F3  F2F7F8F2 00000000 00000000 00000000  *............D4A32782............*    03BAA380
    00C0  00000000 00000000 03BAA170 03BAAB80  00000000 03BAA450 03BAA170 03B97000  *......................u&;.......*    03BAA3A0
    00E0  03BA8000 03B9C1C0 00000000 00490045  00450226 01E50000 00000000 00000001  *......A.............V...........*    03BAA3C0
    0100  03BAB200 03BAA450 03BAA170 00000000  00000000 00000000 00000000 00000000  *......u&;.......................*    03BAA3E0
    0120  00000000 F1F2F3F4 F5F6F7F8 F9F0F1F2  F3F4F5F6 F7F8F9F0 F1F2F3F4 F5F6F7F8  *....1234567890123456789012345678*    03BAA400
    0140  F9F0F1F2 F3F4F5F6 F7F8F9F0 F1F2F3F4  F5F6F7F8 F9F0F1F2 F3F4F5F6 F7F8F9F0  *90123456789012345678901234567890*    03BAA420
    0160  F1F2F3E7 00000000                                                        *123X....                        *    03BAA440
          Status is DTR   Reset,Free,OutB,¬CD,Lose,RA Data abs,RA Resp abs 1
          Connection is using the Conversation at address 00000000 2
          User Data is :1234567890123456789012345678901234567890123456789012345678901234567890123X
 SZDRA 03B9F000 FEPI Connection's RPL 3
    0000  01286EC4 C6C8E2E9 C4C3C400 00000000  04B964F0 000000DD 03B9F000 00000000  *..>DFHSZDCD........0......0.....*    03B9F000
    0020  03BADE60 03B9F450 0000006E 08000000  003C0000 00000000 00000000 00000001  *...-..4&;..>....................*    03B9F020
    0040  00000000 80800000 00000100 00000010  00000000 00000000 00000000 00000000  *................................*    03B9F040
    0060  00000000 00000000 00000000 00000000  000000C5 00010000 00000000 00000000  *...................E............*    03B9F060
    0080  00000000 00000000 08570002 00000000  00000000 00000000 00000000 00000000  *................................*    03B9F080
    00A0  00000000 00000000 00000000 C4F4C1F3  F2F7F8F2 00000000                    *............D4A32782....        *    03B9F0A0
4
 SZDDS 03BAB200 FEPI Connection's Format Extension 5
    0000  00F86EC4 C6C8E2E9 C4C4E200 00000000  04B967B0 000000E1 03BAB200 00000000  *.8>DFHSZDDS.....................*    03BAB200
    0020  00000000 00000000 00000000 00000000  03BB3000 03BB3780 00000000 00000000  *................................*    03BAB220
    0040  00000000 00000000 00000000 00000000  00000000 00000000 00000000 03BAA2E0  *..............................s.*    03BAB240
    0060  00000F00 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03BAB260
    0080  00000780 50185018 50180000 00000000  00000000 07000000 07000000 02004000  *....&;&;&;.................... .*    03BAB280
    00A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03BAB2A0
    00C0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03BAB2C0
    00E0  00000000 00000000 00000000 00000000  00000000 00000000                    *.......................         *    03BAB2E0
          Current Cursor position X'00000000' Current Buffer address X'00000000'
          Screen sizes Current/Default/Alt 024*080/024*080/024*080
 SZDDS.Pgraphic 03BB3000 FEPI Formatted Connection's Graphics Plane
5a
    0000  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03BB3000
    0020 -    077F LINES SAME AS ABOVE                                                                                     03BB3020
 SZDDS.Pattr 03BB3780 FEPI Formatted Connection's Attribute Plane
5a
    0000  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03BB3780
    0020 -    077F LINES SAME AS ABOVE                                                                                     03BB37A0
```

1 Shows the more significant VTAM statuses of the connection.

2 Shows the address of the conversation (if any) currently running over the connection.

3 Shows the connection’s RPL.

4 There is no RESP data for this connection. If there were, it would be shown here.

5 Shows the connection’s formatted data extension.

5a Shows some of the planes of the formatted data extension.

Only some of the format planes are present.

Conversations

```
==SZ.Conv FEPI Active   Conversations 1
 SZDCV 03B9A000 FEPI Active   Conversation
    0000  00686EC4 C6C8E2E9 C4C3E500 00000000  04B96650 000000DF 00000000 00000000  *..>DFHSZDCV........&;...........*    03B9A000
    0020  00000000 00000000 03BA3480 00001000  00000001 00000000 C3C5C3C9 C5F1F7F7  *........................CECIE177*    03B9A020
    0040  0000050C 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03B9A040
    0060  00000000 00000000                                                         *........                        *    03B9A060
Conversation 0000000100000000's internal id is CECIE17700000050 and is in session with the connection at address 03BA3480 2
```

1 Shows the conversation type (active, browse, or inactive).

2 Shows the address (if any) of the connection that is running the conversation, and the internal id of the conversation. The internal id of the conversation is a combination of the CICS transid, termid, and task number for the task running the FEPI request (these can be nulls if there is not a current task).

DQEs

```
==SZ.DQE FEPI API/Expd DQEs
 SZDQE.API/Expd 03B7EDC0 FEPI Work Queue Element 1
    0000  00886EC4 C6C8E2E9 C4D8C500 00000000  0557F7B0 000000E1 00000000 00000000  *.h>DFHSZDQE.......7.............*    03B7EDC0
    0020  00000000 00000000 00000001 80800000  03B7EE48 00000000 00000000 00000000  *................................*    03B7EDE0
    0040  804BB1EB 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03B7EE00
    0060  C3C5C3C9 E2F5F7F6 0000042C 00000000  00000000 00000000 00000318 000001B0  *CECIS576........................*    03B7EE20
    0080  00000000 00000000                                                         *........                        *    03B7EE40
          DQE type is Allocate   ,Internal id is CECIS57600000042 2
          DQE Status is   Post,Normal,NoPRBq,NoIRBq,NoTimr,NoAPI,NoTP8,Finish,  Timed,Stopped,UnFree. 3
 SZDRP 03B7EE48 DQE's API Request Data (DRP) 4
    0000  00B86EC4 C6C8E2E9 C4D9D700 00000000  0557F7B0 000000E1 00000000 00000000  *..>DFHSZDRP.......7.............*    03B7EE48
    0020  00000000 00000001 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03B7EE68
    0040  00000000 00000000 00000168 00000000  D7D6D6D3 C3404040 00000000 00000000  *................POOLC   ........*    03B7EE88
    0060  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03B7EEA8
    0080  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03B7EEC8
    00A0  00000000 00000000 00000000 00000000  00000000 00000000                    *........................        *    03B7EEE8
5
```

1 Shows the type of the DQE.

2 Shows the work the DQE is controlling, and the internal id of the connection on which it is processing.

3 Shows various significant statuses associated with the DQE.

4 Shows the DRP representing the DQE.

5 This DQE does not have any horizontal extensions, nor any associated storage area. If there were any, they would be shown here.

FEPI and VTAM

This section outlines how FEPI interacts with VTAM, and discusses VTAM control blocks and exits.

You should refer to the *OS/390 eNetwork Communications Server: SNA Programming* manual for all information relating to VTAM programming.

VTAM control blocks

FEPI uses standard VTAM programming facilities for its communication. The way in which VTAM control blocks interact with FEPI control blocks is as follows:

ACBs
: Each FEPI node represents a terminal connected to the partner system. Consequently, each node has an **access control block** (ACB). This ACB is opened when the node is acquired, and closed when the node is released.

NIBs
: Each FEPI target contains the applid of the back-end system. This is used to build a **node initialization block** (NIB), when a connection is acquired by issuing a VTAM REQSESS request. In common with CICS data communication, the “confidential” flag is set off.

RPLs
: There are two types of **request parameter list** (RPL) used by FEPI:
- Each FEPI outbound request causes the generation of an RPL. This RPL lasts only for the duration of the FEPI request.
- Each FEPI node has a “Receive-Any” RPL. When an inbound flow occurs, this RPL is attached to the FEPI connection, and turned into a “Receive-Specific” RPL. When the flow has been received, a new “Receive-Any” RPL is generated and attached to the node.

VTAM exits

FEPI communicates with VTAM as asynchronously as possible. Therefore, VTAM exits are extensively used for FEPI communication. The following VTAM exits receive control at specific stages of the communication process:

DFASY
: Processes the receipt of expedited-data-flow control indicators.

LOGON
: Processes the receipt of a CINIT in which FEPI is acting as the primary logical unit (PLU).

LOSTERM
: Processes the loss of a session.

NSEXIT
: Processes:
- The failure of a process that was responded to positively
- A session outage
- The receipt of network service RUs.

SCIP
: Processes the receipt of session-control requests.

TPEND
: Processes the termination of VTAM.

Modules

Module	Function
DFHSZATC	adaptor command tables
DFHSZATR	adaptor program
DFHSZBCL	cleanup API requests at error routine
DFHSZBCS	RM collect statistics
DFHSZBFT	FREE transaction requests scheduler
DFHSZBLO	lost session reporter
DFHSZBRS	RM collect resource ID statistics
DFHSZBSI	signon exit scheduler
DFHSZBST	STSN transaction scheduler
DFHSZBUN	unsolicited data transaction scheduler
DFHSZBUS	RM unsolicited statistics recording
DFHSZDUF	dump formatting routine
DFHSZFRD	formatted 3270 RECEIVE support
DFHSZFSD	formatted 3270 SEND support
DFHSZIDX	SLU P queue install/discard exit
DFHSZPCP	SLU P flow controller
DFHSZPDX	SLU P drain completion exit
DFHSZPID	SLU P send data processor

Front end programming interface (FEPI)

Module	Function
DFHSZPIX	SLU P send completion exit
DFHSZPOA	SLU P send response processor
DFHSZPOD	SLU P receive data processor
DFHSZPOR	SLU P response processor
DFHSZPOX	SLU P receive specific response exit
DFHSZPOY	SLU P receive specific response processor
DFHSZPQS	SLU P REQSESS (request session) issuer
DFHSZPQX	SLU P REQSESS exit
DFHSZPSB	SLU P bind processor
DFHSZPSC	SLU P session controller
DFHSZPSD	SLU P SDT processor
DFHSZPSH	SLU P SHUTC processor
DFHSZPSQ	SLU P quiesce complete (QC) processor
DFHSZPSR	RESETSR processor CSECT
DFHSZPSS	SLU P STSN processor
DFHSZPSX	SLU P OPNSEC completion exit
DFHSZPTE	SLU P TERMSESS processor
DFHSZRCA	node control processor
DFHSZRCT	issue processor
DFHSZRDC	delete connection processor
DFHSZRDG	discard node processor
DFHSZRDN	delete node processor
DFHSZRDP	dispatcher
DFHSZRDS	discard property set processor
DFHSZRDT	discard target procsssor
DFHSZREQ	request passticket module
DFHSZRFC	FREE completion processor
DFHSZRGR	Dispatcher work queue processor
DFHSZRIA	allocate processor
DFHSZRIC	define connection processor
DFHSZRID	discard processor
DFHSZRIF	install free processor
DFHSZRII	install processor
DFHSZRIN	install node processor
DFHSZRIO	ACB open processor
DFHSZRIP	install pool processor
DFHSZRIQ	inquire processor
DFHSZRIS	install processor
DFHSZRIT	install target processor
DFHSZRIW	SET processor
DFHSZRNC	NODE processor

Module	Function
DFHSZRNO	NOOP processor
DFHSZRPM	timer services
DFHSZRPW	request preparation
DFHSZRQR	queue for REQSESS processing
DFHSZRQW	request queue processor
DFHSZRRD	RECEIVE request processor
DFHSZRRT	request release processor
DFHSZRSC	connection processor
DFHSZRSE	SEND request processor
DFHSZRST	START request processor
DFHSZRTM	recovery services
DFHSZRXD	EXTRACT processor
DFHSZRZZ	TERMINATE processor
DFHSZSIP	initialization processor
DFHSZVBN	copy NIB mask to real NIB
DFHSZVGF	get queue element FIFO
DFHSZVQS	REQSESS dispatcher
DFHSZVRA	VTAM receive_any processor
DFHSZVRI	VTAM receive_any issuer
DFHSZVSC	delayed bind processor
DFHSZVSL	SETLOGON request issuer
DFHSZVSQ	VTAM feedback interpreter
DFHSZVSR	VTAM feedback interpreter
DFHSZVSY	VTAM feedback interpreter
DFHSZWSL	RPL exit after SETLOGON
DFHSZXDA	VTAM DFASY exit
DFHSZXFR	RPL exit to free request block
DFHSZXLG	VTAM logon exit
DFHSZXLT	VTAM LOSTERM (lost terminal) exit
DFHSZXNS	VTAM NSEXIT (network services) exit
DFHSZXPM	STIMER IRB exit routine
DFHSZXRA	VTAM RECEIVE_ANY exit
DFHSZXSC	VTAM SCIP (session control) exit
DFHSZXTP	VTAM TPEND exit
DFHSZYLG	RPL exit following logon reject
DFHSZYQR	post for REQSESS processing
DFHSZYRI	VTAM RECEIVE_ANY issuer
DFHSZYSC	VTAM SCIP exit extension
DFHSZYSR	VTAM feedback interpreter
DFHSZYSY	VTAM feedback interpreter
DFHSZZAG	get RECEIVE_ANY request block

Front end programming interface (FEPI)

Module	Function
DFHSZZFR	free RECEIVE_ANY request block
DFHSZZNG	get session control request block
DFHSZZRG	get RPL request block
DFHSZ2CP	SLU2 flow controller
DFHSZ2DX	SLU2 drain completion exit
DFHSZ2ID	SLU2 send data processor
DFHSZ2IX	SLU2 send completion exit
DFHSZ2OA	SLU2 send response processor
DFHSZ2OD	SLU2 receive data processor
DFHSZ2OR	SLU2 response processor
DFHSZ2OX	SLU2 receive specific completion exit
DFHSZ2OY	SLU2 receive specific action module
DFHSZ2QS	SLU2 REQSESS issuer
DFHSZ2QX	SLU2 REQSESS exit
DFHSZ2SB	SLU2 bind processor
DFHSZ2SC	SLU2 session controller
DFHSZ2SD	SLU2 SDT processor
DFHSZ2SH	SLU2 SHUTC processor
DFHSZ2SQ	SLU2 QC processor
DFHSZ2SR	SLU2 RESETSR processor
DFHSZ2SX	SLU2 OPNSEC processor
DFHSZ2TE	SLU2 TERMSESS processor

Chapter 26. Function shipping

Function shipping allows a transaction from one CICS system to access a resource owned by another CICS system.

The CICS function shipping facility enables separate CICS systems to be connected so that a transaction in one system is able to retrieve data from, send data to, or initiate a transaction in, another CICS system. The facility is available to application programs that use the command-level interface of CICS.

Design overview

Figure 56 gives an overview of the function shipping component of CICS.

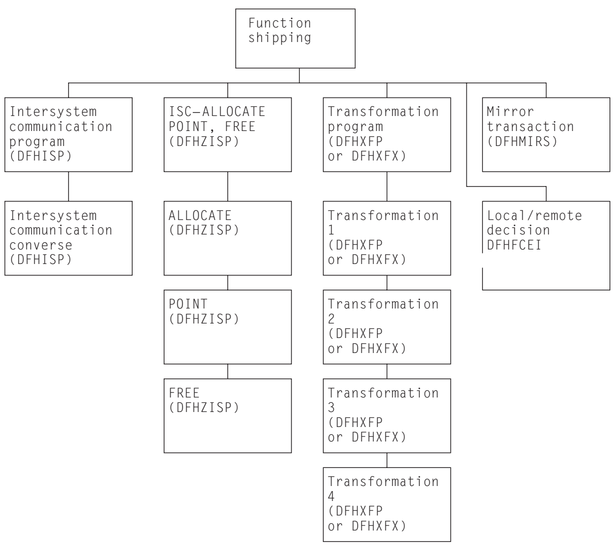


Figure 56. CICS function shipping

This section provides an overview of the operation of CICS when it is being used to communicate with other connected CICS systems for CICS function shipping.

Note: The *CICS Intercommunication Guide* gives a full description of the reasons for CICS function shipping and how the user can take advantage of the facility.

Application programming functions with CICS function shipping

The functions provided by CICS are extended for CICS function shipping so that an application program can issue the following types of command and have them executed on another system:
- Temporary-storage commands
- Transient data commands
- Interval control commands
- File control commands
- DL/I calls
- Program link commands (DPL).

Application programs can use these extended functions without having to know where the resources are actually located; information about where resources are located is contained in the appropriate tables prepared by the system programmer. Alternatively, provision is made for an application program to name a remote system explicitly for a particular request.

Support for syncpoints, whether explicit (through EXEC CICS SYNCPOINT commands) or implicit (through DL/I TERM calls), allows updates to be made in several systems as part of a single logical unit of work.

Error handling routines may need to be extended to handle additional error codes that may be returned from a remote system. See the *CICS Intercommunication Guide* for the relevant conditions.

Local and remote names

For a transaction to access a resource (such as a file or transient data destination) in a remote system, it is usually necessary for the local resource table to contain an entry for the remote resource. The name of this entry (that is, the name by which the resource is known in the local system) must be unique within the local system. The entry also contains the identity (SYSIDNT) of the remote system and, optionally, a name by which the resource is known in the remote system. (If this latter value is omitted, it is assumed that the name of the resource in the remote system is the same as the name by which it is known in the local system.)

Mirror transactions

When a transaction issues a command for a function to be executed on a remote system, the local CICS system encodes the request and sends it to the system identified in the appropriate CICS table, or on the command itself. The receipt of this request at the remote system results in the attachment of one of the CICS-supplied mirror transactions, namely, CSMI, CSM1, CSM2, CSM3, and CSM5, or transactions CVMI and CPMI. All these transactions use the mirror program, DFHMIRS. (CVMI services LU6.2 sync level 1 requests, including those from CICS/VM, and CPMI services function shipping from CICS OS/2.)

For distributed program link (DPL) requests shipped from a CICS application region to a CICS resource region, the name of the mirror transaction to be attached may be specified by the user. If you specify your own mirror transaction, you must define the transaction in the resource region and associate it with the CICS-supplied mirror program, DFHMIRS.

The CVMI and CPMI transactions service requests sent as part of an LU6.2 synclevel 1 conversation, unlike the other transactions that service requests sent as part of an LU6.2 synclevel 2 conversation or an MRO or LU6.1 conversation.

A mirror transaction executes the initiating transaction's request and reflects back to the local system the response code and any control fields and data that are associated with the request. If the execution of the request causes the mirror transaction to abend, this information is also reflected back to the initiating transaction.

If a resource has browse place holders or is recoverable, or the lock has been acquired, the mirror transaction becomes a **long-running mirror** and does not end until the issuing transaction ends the logical unit of work (that is, a SYCNPOINT or RETURN). Any resources the mirror has acquired are freed when the initiating transaction issues the appropriate command to free those resources.

Initialization of CICS for CICS function shipping

If CICS has been generated with the appropriate options for intercommunication, the initialization of CICS with the ISC=YES system initialization parameter specified causes the following modules to be loaded:
- DFHISP (intersystem communication program)
- DFHXFP (data transformation program)
- DFHXFX (optimized data transformation program).

The entry point addresses of these modules are contained in the optional features list, which is addressed by CSAOPFLA in the CSA.

The mirror program, DFHMIRS, is not loaded until a request is received from a remote system. (This program can only be loaded if there is an associated PPT entry ***and*** PCT entries for mirror transactions CSMI, CSM1, CSM2, CSM3, and CSM5 or for transactions CVMI and CPMI; sample entries are created by the CSD group DFHISC.)

Note: The ISC=YES system initialization parameter causes other modules besides those specified earlier to be loaded; the ones mentioned here are those specifically required for CICS function shipping.

Communication with a remote system

For multiregion operation, communication between CICS systems can be implemented:

- Through support in CICS terminal control management modules and by use of a CICS-supplied interregion program (DFHIRP) loaded in the link pack area (LPA) of MVS. DFHIRP is invoked by a type 3 supervisory call (SVC). The SVC moves the data to an intermediate area in key 0 MVS CSA storage, and schedules an SRB to move the data from the intermediate area to the target.
- By MVS cross-memory services (DFHXMP), which you can select as an alternative to the CICS type 3 SVC mechanism. Here, DFHIRP is used only to open and close the interregion links. Cross-memory services do not require intermediate MVS CSA storage areas.
- By the cross-system coupling facility (XCF) of MVS. XCF is required for MRO links between CICS regions in different MVS images of an MVS sysplex. It is selected dynamically by CICS for such links, if available.

For ISC, communication between CICS systems takes place via ACF/VTAM links. CICS and the CICS application programmer are independent of, and unaware of, the type of physical connection used by ACF/VTAM to connect the two systems.

Protocols

Requests and replies exchanged between systems for CICS interval control, CICS transient data, CICS temporary storage, and DL/I functions are shipped using the standard protocol as defined for SNA logical unit type 6.1.

Requests and replies for CICS file control functions are shipped using a private protocol (with function management headers of type 43).

Symmetrical bracket protocol

Logical unit type 6.1 (LU6.1) sessions between two CICS systems require most protocols to be symmetrical; therefore, CICS receives (as well as sends) end bracket.

Shutdown protocol

The LU6.1 shutdown protocol does not use the SHUTDOWN command; it uses the data flow control commands SBI (stop bracket initiation) and BIS (bracket initial stopped). Shutdown is executed as part of session termination (by DFHZCLS) and ensures that, when a session is terminated normally (as a result of a master terminal release command or a normal CICS shutdown), there are no unfinished syncpoint requests on the session. This means that when the session is initiated, no resynchronization sequence is required.

Sender error recovery protocol (ERP)

CICS support for LU6.1 uses a symmetrical SNA protocol called **Sender ERP**. In addition, when CICS wishes to send a negative response to a remote system, it sends a special negative response (0846), which indicates that an ERP message is to follow. This ERP message contains the real system and user sense values, together with a text message. The negative response and ERP message are built by DFHZEMW, and are received and processed by DFHZRAC, DFHZRVX, and DFHZNAC.

Resynchronization protocol

CICS support for LU6.1 sessions that use the syncpoint protocol has associated resynchronization logic, which is used during the initiation of a session after a previous session has terminated abnormally. This logic is used to generate messages concerning the outcome of any logical units of work that were **in doubt** when the previous session failed. The modules involved are DFHZRSY, DFHZSCX, and DFHZNAC.

CICS function shipping environment

This section describes the system entries for function shipping in the terminal control table, and how function shipping requests or replies are transformed between the format suitable for transmission and the internal parameter list format.

System entries in the terminal control table

All remote systems with which a given system is able to communicate are identified and described in terminal control table system entries (TCTSEs). The name of the system entry is the name specified in the SYSIDNT field of the CICS table entry describing a remote resource.

CICS uses the TCTSE as an anchor point to queue requests made by CICS transactions for connection to the remote system.

Figure 57 shows three TCTTEs. If a transaction fails and you get a transaction dump, this figure shows you how to find the relevant TCTTEs from the TCA.

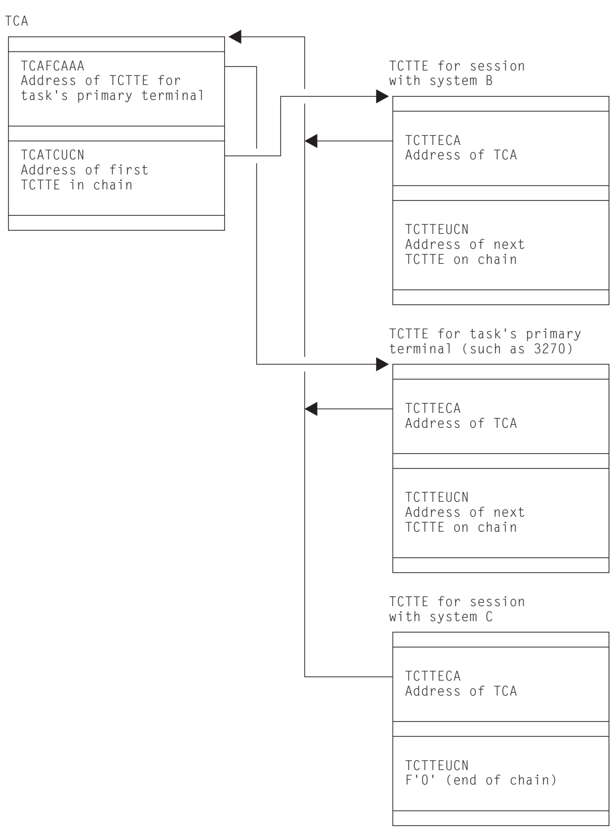


Figure 57. Task's view of CICS function shipping TCTTEs

Transformation of requests and replies for transmission between systems

Before a request or reply can be transmitted, it must be transformed from its internal, parameter list (EXEC interface) format to a format suitable for transmission; when received after transmission, the request must be transformed back into a parameter list format.

There are four such transformations (numbered 1 through 4), which are performed by DFHXFP, or by DFHXFX if optimized data transformations are possible. The latter only applies to data transformations for function shipping in an MRO environment, excluding those relating to DL/I requests.

Transformation 1
For a request to be sent by the originating system; transforms from parameter list format to transmission format.

Transformation 2
For a request received by the mirror transaction; transforms from transmission format to parameter list format.

Transformation 3
For a reply to be sent by the mirror transaction; transforms from parameter list format to transmission format.

Transformation 4
For a reply received by the originating system; transforms from transmission format to parameter list format.

The parameter list format above refers to the parameter list that is normally passed to DFHEIP (for CICS requests) or to DFHDLI (for DL/I requests).

The transmission formats of these requests and replies (excluding those for syncpoint protocol) are described in the DFHFMHDS DSECT.

Information that DFHXFP and DFHXFX need to retain between transformations 1 and 4 (in the originating system) or between transformations 2 and 3 (in the mirror system) is stored in a transformer storage area called XFRDS; See*CICS Data Areas* for a detailed description of this control block.

CICS function shipping—handling of EXEC CICS commands

This section describes the sending and receiving of requests and replies (other than DL/I or syncpoint requests) between two connected systems at the **application-layer** level; see Figure 58 on page 282. (The **function management** and **data flow control** layers, implemented by CICS terminal control, work in the same way, regardless of the type of request being transmitted.)

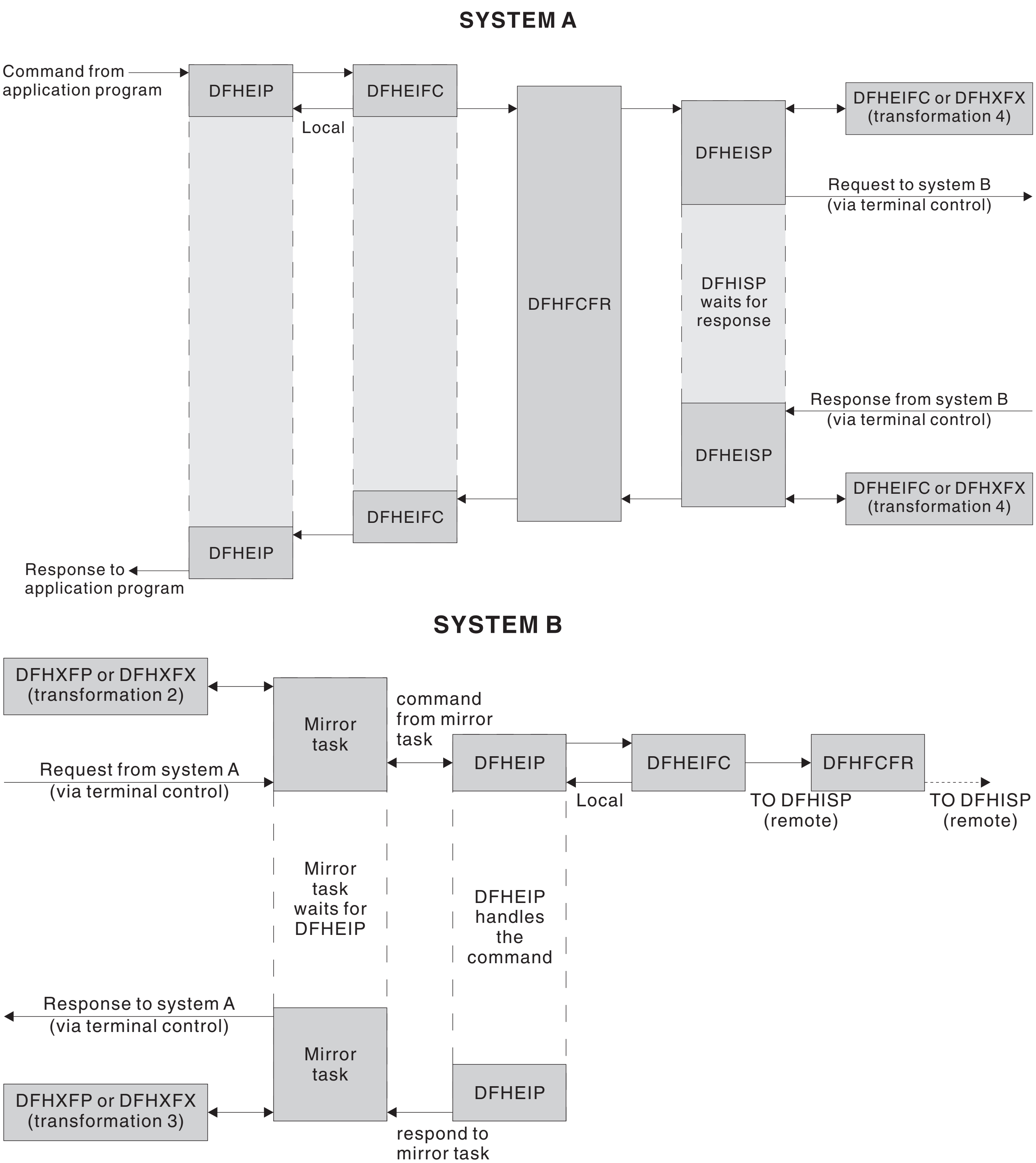


Figure 58. Overview of CICS function shipping

Sending a request to a remote system

A CICS command is handled for an application program by the EXEC interface program, DFHEIP. DFHEIP analyzes the arguments of each statement to determine the requested function and to assign values into the appropriate CICS control blocks; DFHEIP also performs storage control and error checking on behalf of the application programmer.

If the system has been initialized with the ISC=YES system initialization parameter, and if the request is for one of the functions that could be executed on a remote system (see “Application programming functions with CICS function shipping” on page 277), DFHEIP invokes a local/remote decision routine, which inspects the appropriate CICS table to determine whether the request is for a local or a remote resource (unless a remote system has specifically been requested). For all requests except file control, this local/remote decision is taken in DFHEIP. For file control requests, the decision is taken in the file control function shipping interface module, DFHFCRF (see Chapter 24, “File control,” on page 163).

If the resource is local:

- DFHEIP invokes the appropriate EXEC interface processor module to process the request locally.
- DFHEIFC calls the file control file request handler, DFHFCFR, to process the request locally, and finally returns control to DFHEIP.

Note: A SYSID value that names the local system also causes the request to be processed locally.

If the resource is remote, DFHEIP or DFHFCRF:

1. Allocates a transformer storage area (XFRDS) chained off the EXEC interface storage EIS. XFRDS provides a central area in which all information about processing of the request can be accessed.
2. Places the following data in XFRDS:
   - Name of remote system, for subsequent use by DFHISP (in XFRDS field XFRSYSNM)
   - Address of the application’s list of parameters (EXEC parameter list) associated with the command being executed (in XFRDS field XFRPLIST)
   - Address of the table (FCT, if DFHFCRF; DCT, and so on, otherwise) for the requested resource (in XFRDS field XFRATABN).
3. Issues a DFHIS TYPE=CONVERSE macro, which passes control to the CICS function shipping program DFHISP.

DFHISP obtains the address of the TCTSE for the remote system and places it in XFRDS field XFRATCSE. DFHISP obtains the address of the TCTTE that controls the session with the remote system and places it in XFRDS field XFRATCTE. (DFHISP obtains the address by issuing a DFHTC TYPE=POINT macro. If no session is established, there is no TCTTE; in this case DFHISP issues a DFHTC TYPE=ALLOCATE macro to establish the session TCTTE.)

If no session can be allocated because, for example, all sessions are out of service, DFHISP determines whether or not the function request can be queued for shipping at a later time. If it the request can be queued, then XFRATCTE is set to zero.

Optionally (if a TIOA already exists from an earlier CICS function shipping request from the same application), DFHISP also places the address of the TIOA in XFRDS field XFRATIOA.

DFHISP then invokes DFHXFP, or DFHXFX for optimized transformations, to transform the requested command and parameter list into a form suitable for transmission. This is known as **transformation 1**, which:

1. Transforms the original **command** into an appropriate type of request for transmission.
2. Converts the EXEC parameter list into a **data unit** having a standardized character-string format (together with a function control header) suitable for transmission. The data unit is built in the TIOA and contains a copy of each of the parameters that are addressed by the EXEC parameter list. (For economy of transmission, certain types of data are compressed before being placed in the TIOA.)
3. Returns control to DFHISP.

Note: If local queuing is in effect, the data unit is built in user storage.

DFHISP then invokes terminal control to transmit the contents of the TIOA to the remote system and waits for the reply from the remote system, if necessary.

If local queuing is in effect, DFHISP issues a DFHIC TYPE=PUT macro specifying transaction CMPX, which sends the data unit at a later time.

Receiving a request at a remote system

Terminal control receives the request transmission and attaches one of the mirror transactions.

The mirror program allocates space for XFRDS in its LIFO storage area. As in the requesting system, XFRDS is a central area in which all information about the processing of the received request can be accessed. The mirror program places the following data in XFRDS:

- Address of the session TCTTE (in XFRDS field XFRATCTE)
- Address of the TIOA (in XFRDS field XFRATIOA).

The mirror program also allocates scratch pad storage in the LIFO storage area for use by DFHXFP (or DFHXFX) in building argument lists. The address of this storage is placed in XFRPLIST.

The mirror program then invokes DFHXFP, or DFHXFX for optimized transformations, to transform the received request into a form suitable for execution by DFHEIP. This is known as **transformation 2**, which:

1. Transforms the received request (as coded in the function management header of the data unit) into an appropriate CICS command.
2. Decodes the TIOA and builds (in the ***first*** part of the STORAGE area) an EXEC parameter list that basically consists of addresses that point to fields in the TIOA. (Those fields that were compressed for transmission are expanded and placed in the ***second*** part of the STORAGE area; for these fields, the EXEC parameter list points to the expanded versions, not the compressed versions in the TIOA.)

   Note: The NOHANDLE option is specified on each EXEC CICS command that is created; this has the effect of suppressing DFHEIP's branching to an error routine.
3. Returns control to the mirror program.

The mirror program then invokes DFHEIP (in the same way as for an application program), passing to it (in register 1) the address of the EXEC parameter list just built.

DFHEIP or DFHFCRF determines whether the request is for a remote resource on yet another system or for a local resource. If the resource is remote, DFHEIP or DFHFCRF allocates a new and separate transfer storage area XFRDS and invokes DFHISP (as described under "Sending a request to a remote system" on page 282).

If the resource is local, the reply is processed for the mirror program in the usual way.

Sending a reply at a remote system

The process of sending a reply in response to a request from another system is similar to that for sending a request; see "Sending a request to a remote system" on page 282.

When DFHEIP has successfully completed execution of the command, control is returned to the mirror program with the results of the execution in the EXEC interface block (EIB). The mirror program then invokes DFHXFP, or DFHXFX for optimized transformations, to transform the command response into a suitable form for the transmission of the reply. This is known as **transformation 3**, which:

1. Checks whether the existing TIOA is long enough to take the reply; if not, DFHXFP (or DFHXFX) frees the existing TIOA and creates a new one.
2. Converts the EXEC parameter list (kept in the scratch pad area STORAGE) into a **data unit** having the standardized character-string format suitable for transmission. The data unit is built in the TIOA. If the request is received by the mirror program without CD (that is, the requesting system did not expect

a reply), the mirror program issues a DFHTC TYPE=READ or TYPE=FREE macro. If an error is detected, the mirror program is forced to abend, so that at least a record of the request failure is written.

3. Returns control to the mirror program.

The mirror program then invokes terminal control to transmit the TIOA. (The mirror program does this by issuing a DFHTC TYPE=(WRITE,WAIT,READ) macro if the mirror program holds any state information that must be held for a further request or until a syncpoint. Otherwise, a DFHTC TYPE=(WRITE,LAST) macro is issued.

Receiving a reply from a remote system

Terminal control receives the reply and returns control to the initiating task; in particular, control is passed to DFHISP, which has been waiting for the reply.

DFHISP invokes DFHXFP, or DFHXFX for optimized transformations, (passing to it the address of the XFRDS area) in order to transform the reply into the form expected by the application program. This is known as **transformation 4**, which:

1. Obtains the addresses of the TIOA and of the original EXEC parameter list from XFRATIOA and XFRPLIST in the XFRDS area.
2. Uses data in the reply to complete the execution of the original command. For example:
   - Sets return codes in the EIB from status bits in the reply
   - Stores other received data (if any) in locations specified in the original EXEC parameter list.
3. Frees the TIOA.
4. Returns control to DFHISP.

DFHISP returns control to DFHEIP (if appropriate through DFHEIFC), which raises any error conditions associated with return codes set in the EIB. DFHEIP then returns control to the application program.

CICS function shipping—handling of DL/I requests

DL/I requests are handled in a similar manner to that for CICS commands; see Figure 59 on page 286.

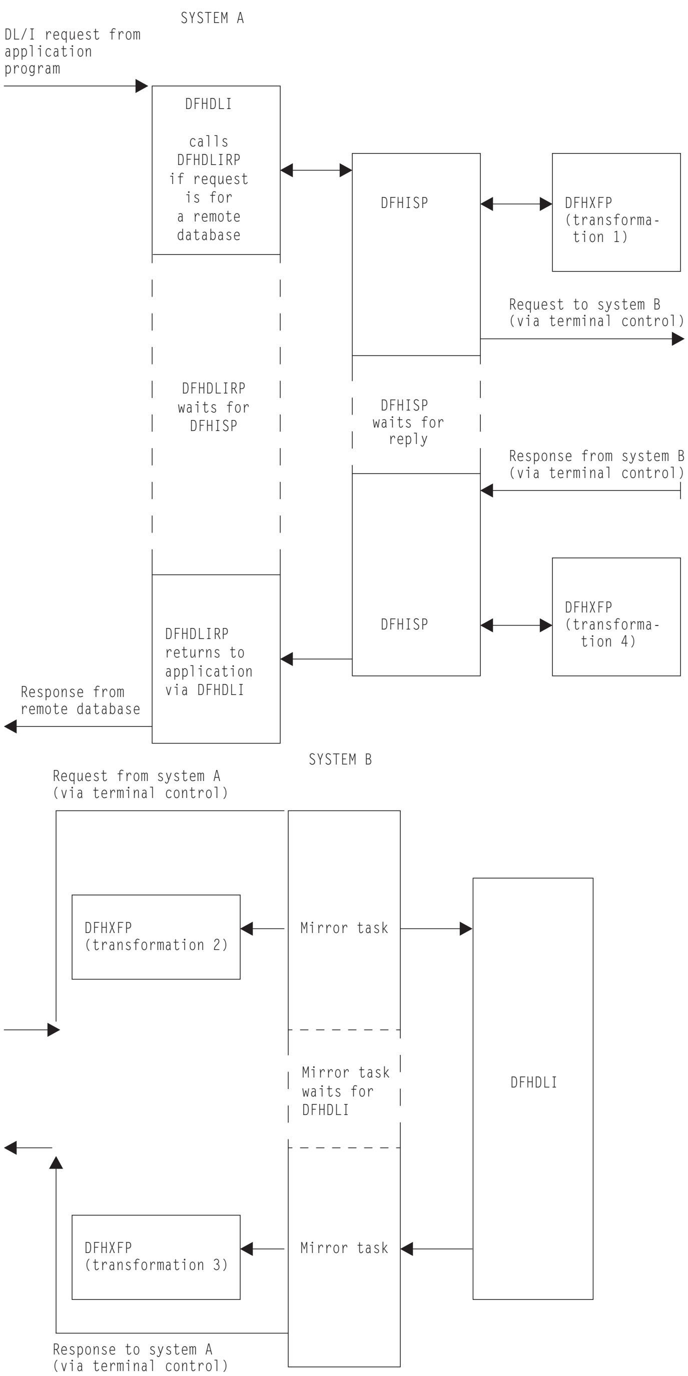


Figure 59. Overview of CICS function shipping of DL/I requests

Sending a DL/I request to a remote system

All DL/I requests are handled by DFHDLI.

DFHDLI determines whether the request is for a remote, or DBCTL database, and routes the request to the appropriate DL/I call processor. If the request is for a remote database, DFHDLI invokes DFHDLIRP, which passes control to DFHISP by issuing a DFHIS TYPE=CONVERSE macro.

DFHISP then:
1. Invokes DFHXFP to transform the request into a form suitable for transmission
2. Invokes terminal control to transmit the request.

Receiving a DL/I request at a remote system

As for a CICS request, the appropriate mirror transaction (in this case, CSM5) is attached.

The mirror program invokes DFHXFP to transform the received request into a form suitable for execution by DFHDLI.

The mirror program then passes the request to DFHDLI in the same way as any other application program would. DFHDLI determines what type of DL/I request is being made and then routes the request to the appropriate DL/I call processor: DFHDLIRP (remote, that is, daisy-chained to yet another system), or DFHDLIDP (DBCTL).

Sending a DL/I reply at a remote system

When DFHDLI has successfully completed the request, control is returned to the mirror program with the results in the user interface block (UIB). The mirror program then:
1. Invokes DFHXFP to transform the results into a form suitable for transmission
2. Invokes terminal control to transmit the reply.

Receiving a DL/I reply from a remote system

On receipt of the reply, terminal control returns control to DFHISP, which has been waiting for the reply; DFHISP then invokes DFHXFP to transform the reply into a form that can be used by DFHDLI. DFHXFP sets the return codes in an intermediate control block, DFHDRX, so that they can ultimately be copied to the UIB or the TCA for the application program. Control is then returned from DFHISP through DFHDLIRP to DFHDLI, and finally back to the application program.

Terminal control support for CICS function shipping

Terminal control support for CICS function shipping falls into the following three main areas:
1. TCTTE allocation functions, ALLOCATE, POINT, and FREE. These functions are used mainly by DFHISP to allow a CICS transaction to own additional TCTTEs. These are session TCTTEs to remote systems; these functions are supported by DFHZISP.
2. Syncpoint functions, SPR, COMMIT, ABORT, and PREPARE. These functions are used by the recovery manager connectors to implement the syncpoint protocol; these functions are supported by DFHZIS1.
3. LU6.1 functions. These functions are used by users of terminal control to support the data flow control protocols used in a LU6.1 session.

The functions described in areas 1 and 2 above are extensions to the DFHTC macro that are intended for internal use by CICS control programs only; they are not documented in the user manuals.

TCTTE allocation functions

Terminal control provides the following TCTTE-related functions:

ALLOCATE function

This allocates to the requesting transaction a session TCTTE for communication with a remote system. The name of the remote system is passed as a parameter. The address of the allocated TCTTE or a return code is returned to the requester. DFHZISP uses the DFHZCP automatic transaction initiation (ATI) mechanism to allocate the session.

If the allocation request cannot be satisfied immediately, an automatic initiate descriptor (AID) is created, and is chained off the system entry; the AID is used to remember, and subsequently to process, the outstanding allocation request.

Parallel sessions can be allocated explicitly, or implicitly by reference to a remote resource; sessions are automatically initiated at allocation time, if necessary. They can also be initiated by a master terminal ACQUIRE command, or automatically during CICS initialization if CONNECT=AUTO is specified in the TCTTE.

POINT function
This causes terminal control to supply the requesting task with the address of a session TCTTE for a named remote system. The TCTTE must have been previously allocated to the requesting task.

FREE function
This detaches a TCTTE from the owning task and makes it available for allocation to another transaction. (The FREE function is the opposite of the ALLOCATE function.)

TERM=YES operand
This operand enables the issuer of a terminal control macro to select explicitly the TCTTE to which the requested function is to be applied. The address of the TCTTE to be processed is passed as a parameter of the request; the TCTTE must have been previously allocated to the requesting task.

FREE TCTTE indicator
This indicator is set as a result of the remote system issuing a (WRITE,LAST) or FREE request to show that the current conversation has finished and that the session should be freed by the current owner of the TCTTE. The receiver of the FREE indicator (usually DFHISP) must issue a FREE request.

Syncpoint functions

For ISC, terminal control provides the following syncpoint functions (the equivalent functions for IRC are provided by DFHZIS1):

SPR (syncpoint request) function
This request is issued by the recovery manager connector during syncpoint processing, and causes terminal control (DFHZSDR) to send a request that has a definite DR2 response requested. This tells the other side of the session that a syncpoint is required.

COMMIT function
This request is issued by the recovery manager connector when syncpoint has been completed. It causes a positive DR2 response to be sent, signaling the successful completion of syncpoint protocol.

ABORT function
This request causes either a negative DR2 response or an LUSTATUS command to be sent, indicating that a requested syncpoint operation could not be completed successfully, or that there has been an abnormal end of the current logical unit of work.

PREPARE function
This request causes an LUSTATUS command to be sent to the mirror in the remote system and indicates that a syncpoint should be taken.

VTAM secondary half-session support

CICS acts as both the primary and the secondary halves of an LUTYPE6 session. To implement secondary half-session support, CICS VTAM terminal control has to do two things:

1. Implement the secondary half of the data flow control and session control protocols that CICS already uses as a primary.
2. Use the secondary API provided by VTAM.

The terminal control functions provided by CICS are independent of primary/secondary considerations. Differences between the primary and secondary VTAM interfaces are contained within the CICS modules that issue the appropriate VTAM request. The secondary support functions appear principally in the DFHZCP modules shown in Table 9 on page 289.

Table 9. VTAM secondary support functions

Modules	Function	Secondary function
DFHZSIM	Request LOGON	Use REQSESS macro
DFHZOPN	OPNDST	Use OPNSEC macro
DFHZSCX	SCIP exit	Receive and process BIND, STSN, SDT, CLEAR, and UNBIND commands
DFHZCLS	CLSDST	Use TERMSESS macro
DFHZRSY	Resynchronization	Build STSN responses
DFHZSKR	Respond to	Send responses to BIND, SDT, and STSN commands
DFHZRAC, DFHZRVX	Receive	Receive and process BID commands
DFHZATI, DFHZRVX, DFHZRAC	Bracket protocol	Implement secondary contention resolution using bracket protocol
DFHZNSP	Network services error exit	Handle secondary LOSTERM type of errors

NOCHECK option function handling

The transmission of a START NOCHECK command and associated data is handled in a slightly different manner from that for other CICS function shipping commands. Compared with the process described earlier in “Security manager domain’s generic gates” on page 1339, the major differences are:

- After DFHISP has allocated the session TCTTE to the requesting task, the transformation program DFHXFP (or DFHXFX) performs **transformation 1**. In addition, the transformation program detects that a START NOCHECK command is being processed and passes this fact to DFHISP in its return code. Accordingly, DFHISP issues a DFHTC TYPE=WRITE macro, which is deferred until syncpoint, return, or another function-shipped request on that ISC session.
- DFHISP returns to its caller.
- On the receiving system, DFHEIP handles the START NOCHECK command in the usual way and then terminates when the command has been executed; no response is sent back to the first system.

Exits

There are two global user exit points in DFHISP: XISCONA and XISLCLQ. For further information about using these exit points, see the *CICS Customization Guide*.

Trace

The following point ID is provided for the intersystem program:
- AP 00DF, for which the trace level is IS 1.

The following point IDs are provided for function shipping data transformation:
- AP D9xx, for which the trace level is IS 1.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 27. “Good morning” message program

The CICS good morning program issues a “good morning” message for VTAM logical units.

Design overview

This module is invoked by running the CSGM system transaction.

If a satisfactory OPNDST has occurred (detected in the OPNDST exit, DFHZOPX) and if a “good morning” message has been requested on the TCT TYPE=TERMINAL entry, an NACP request is queued. NACP issues a DFHIC TYPE=INITIATE for this transaction.

This module determines the terminal type, sets up the appropriate control characters, gets a TIOA, and writes the message.

For a 3270 terminal, if the operator has entered data before the message has been received, NACP may be invoked to handle intervention required. In this case the transaction is abended and the write operation terminated.

A default message text is generated by DFHTCTPX and can be overridden by an option on the TCT TYPE=INITIAL statement. The text is stored in the TCT prefix.

Modules

DFHGMM

Exits

The XGMTEXT global user exit point is provided in DFHGMM. For further information about this, see the *CICS Customization Guide*.

Trace

No trace points are provided for this function.

Chapter 28. Interregion communication (IRC)

CICS multiregion operation (MRO) enables CICS regions that are running in the same MVS image, or in the same MVS sysplex, to communicate with each other. MRO does not support communication between a CICS system and a non-CICS system such as IMS. [2]

ACF/VTAM and SNA networking facilities are not required for MRO. The support within CICS that enables region-to-region communication is called **interregion communication** (**IRC**). IRC can be implemented in three ways:

- Through support in CICS terminal control management modules and by use of a CICS-supplied interregion program, DFHIRP, loaded in the MVS link pack area. DFHIRP is invoked by a type 3 supervisory call (SVC).
- By MVS cross-memory services, which you can select as an alternative to the CICS type 3 SVC mechanism. Here, DFHIRP is used only to open and close the interregion links.
- By the cross-system coupling facility (XCF) of MVS. XCF is required for MRO links between CICS regions in different MVS images of an MVS sysplex. It is selected dynamically by CICS for such links, if available.

This section describes the communication part of MRO. Chapter 35, “Multiregion operation (MRO),” on page 325 gives a brief description of multiregion operation.

Design overview

For information about the design and implementation of interregion communication facilities, and about the benefits of cross-system MRO, see the *CICS Intercommunication Guide*.

Control blocks

IRC uses two levels of control blocks:

1. A CICS/MRO terminal control layer
2. An interregion SVC layer interfaced by the DFHIR macro.

Terminal control layer

The CICS/MRO terminal control layer is shown in Figure 60 on page 294.

This layer uses the cross-region block (CRB). This is a global (that is, one per CICS system) block that is created in the CICS dynamic storage area above the 16MB line (the ECDSA) when IRC is initialized, and provides information to communicate with the IRC SVC. See Figure 61 on page 296.

2. The external CICS interface (EXCI) uses a specialized form of MRO link to support: communication between MVS batch programs and CICS; DCE remote procedure calls to CICS programs.

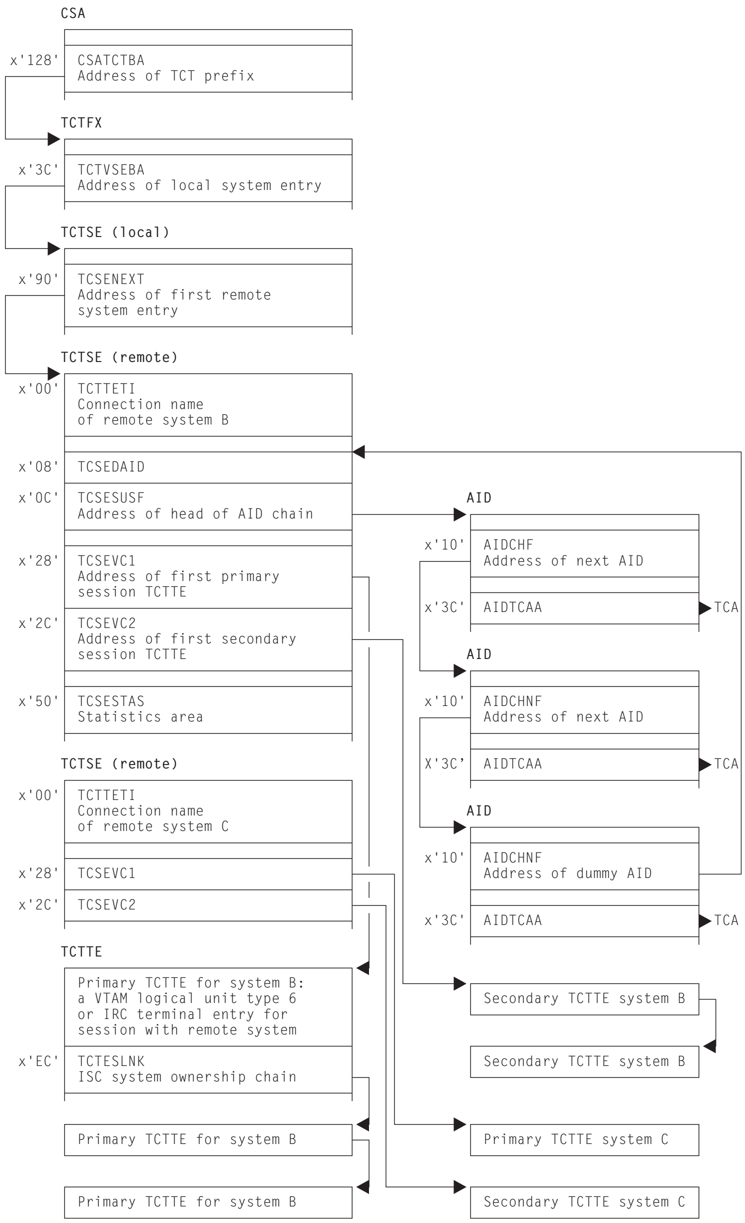


Figure 60. CICS/MRO terminal control layer of control blocks (Part 1 of 2)

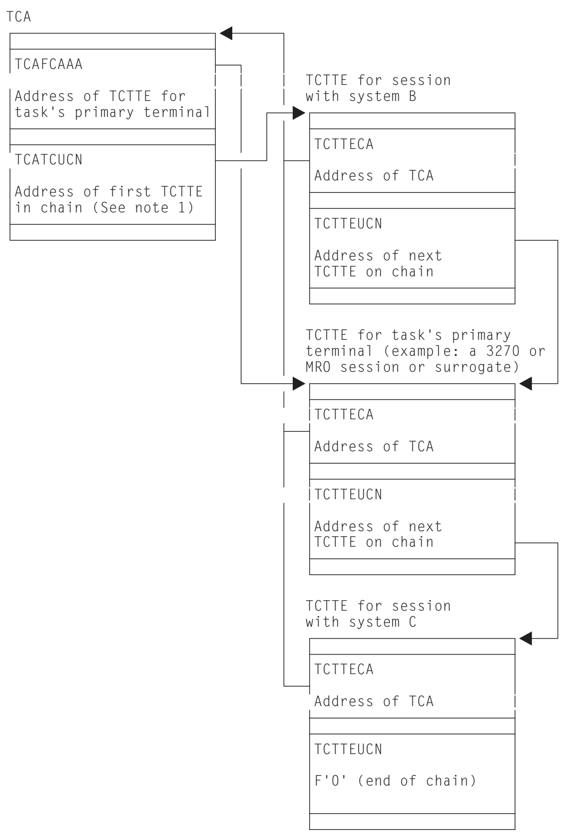


Figure 60. CICS/MRO terminal control layer of control blocks (Part 2 of 2)

Notes:

1. The first TCTTE on the chain is not necessarily the TCTTE for the task's primary terminal.
2. A task has allocated MRO sessions to other systems.
3. TCTTEs are described more fully in Chapter 56, "Terminal control," on page 405.
4. Primary TCTTEs relate to Receive sessions, and secondary TCTTEs relate to Send sessions.
5. TCSEVC1 is the label on the address of the TCTTE of the first primary session. TCSEVC2 is that of the first secondary session.
6. The primary and secondary sessions each have sets of TCTTEs. These are found by using the DFHTC CTYPE=LOCATE macro.
7. A TCTTE is allocated for a surrogate session in transaction routing.

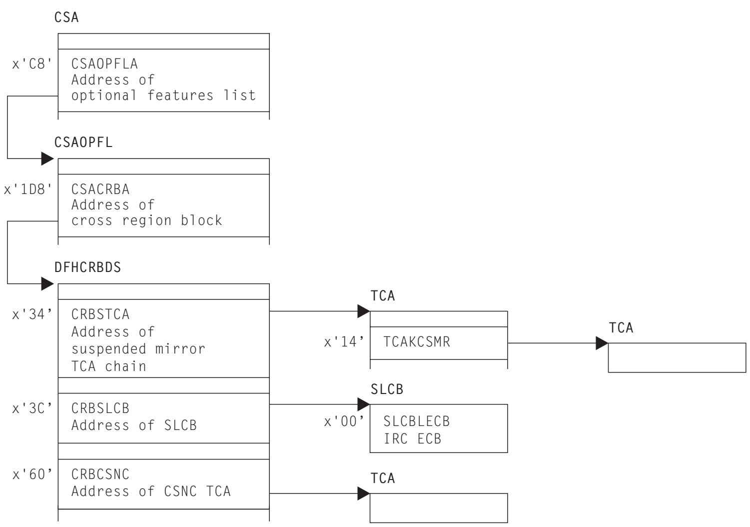


Figure 61. Cross-region block (CRB)

DFHIR layer

The interregion SVC layer interfaced by the DFHIR macro is shown in Figure 62.

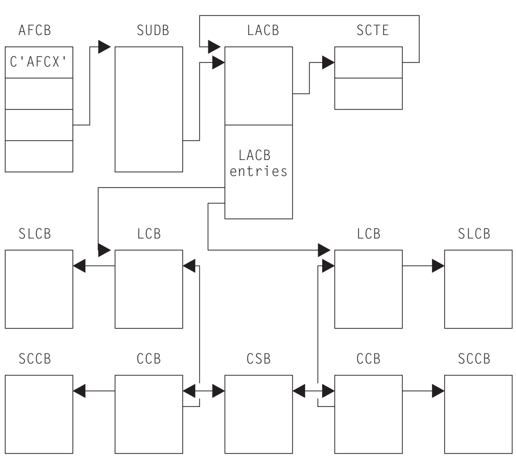


Figure 62. Interregion SVC layer of control blocks interfaced by the DFHIR macro

This layer uses the following control blocks, which, unless otherwise stated, reside in subpool 241 in MVS storage:

- Global (that is, one per MVS system) housekeeping (used by DFHIRP)

  Subsystem control table extension (SCTE)

  The SCTE is dynamically created, and contains information about the number of regions logged on to DFHIRP. It is used to locate the LACB. See also Figure 74 on page 361, which shows the subsystem interface control blocks, including a pointer to the SCTE in the CICS subsystem anchor block (SAB).

  Logon address control block (LACB)

  The LACB contains entries to identify the regions that have logged on, and contains the address of the region's logon control block (LCB).

- Local housekeeping (used by DFHIRP)

Logon control block (LCB) The LCB is created for each successful log on.

Logon control block entry (LCBE)
The LCBE contains the basic control information for each IRC system with which this system communicates. It addresses the connection control blocks (CCBs).

Subsystem user definition block (SUDB)
A SUDB provides access to IRC control blocks. There is one SUDB for each TCB that is currently logged on (so each SUDB may have multiple LCBs associated with it). The SUDB contains TCB-related data and working storage.

Connection control block (CCB)
A CCB is created for each IRC send-receive session, and contains information controlling the connection to the other region. When the connection is in use, it addresses the CSB.

Connection status block (CSB)
The CSB provides status information about the connection between two regions.

MVS transfer buffers (MVS SRB mode)
The MVS transfer buffers are used to transfer IRC data between regions, and reside in subpool 231 in MVS storage.

Terminal control layer and DFHIR layer

Figure 63 shows the control blocks that are accessed by both the terminal control layer and the DFHIR layer. Figure 64 on page 298 shows the location of these control blocks in MVS virtual storage.

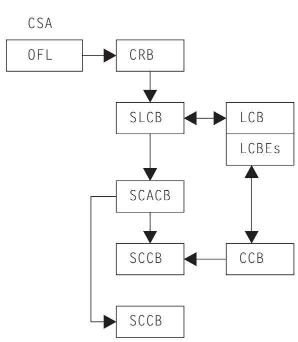


Figure 63. Control blocks accessed by CICS/MRO terminal-control layer of control blocks and by interregion SVC layer of control blocks

The following blocks are used by both the terminal control layer and the DFHIR layer. These blocks are allocated at logon time within a single MVS GETMAIN, and, unless otherwise stated, reside in subpool 251 of MVS storage.

Subsystem logon control block (SLCB)
The SLCB is used by the IRC SVC and region and contains the master ECB, posted when the region has IRC activity. It is pointed to by the CRB and LCB.

Subsystem connection address control block (SCACB)
The SCACB contains entries allowing the addressing of SCCBs from the SLCB.

Subsystem connection control block (SCCB)
The SCCB is created for each IRC send-receive session, and is allocated at logon. It contains the ECB, posted when input for the session is available.

Note: There is a one-to-one relationship between TCTTEs and SCCBs when they are in use.

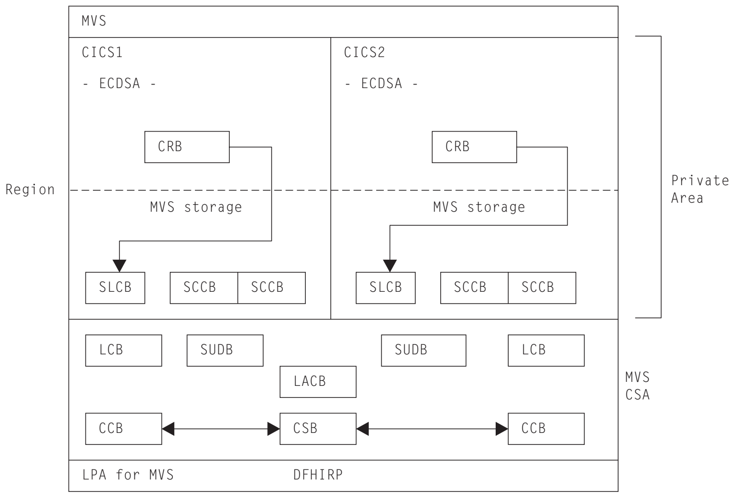


Figure 64. Location of control blocks in MVS virtual storage

MRO ECB summary

The following is a summary of the MRO event control blocks (ECBs):

Name	Location	Who waits	Who posts
Dependent ECB	SCCB	Application (TC WAIT)	DFHIRP
LOGON ECB	SLCB	CICS (KCP, Op sys WAIT list)	DFHIRP
Link ECB	LCB	DFHIRP (Op sys WAIT)	DFHIRP
Work queue ECB	QUEUE	CSNC transaction	DFHIRP DFHZIS2 DFHZLOC

See the *CICS Data Areas* manual for a detailed description of the CICS control blocks.

Modules

Figure 65 gives an overview of the modules involved with interregion communication.

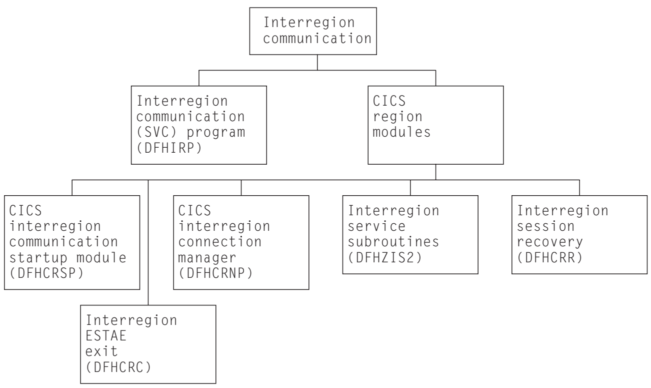


Figure 65. Interregion communication

The modules for IRC are of two types:
1. The interregion communication programs: DFHIRP and DFHXMP.
2. CICS address space modules: DFHCRC (interregion ESTAE exit), DFHCRNP (CICS interregion connection manager), DFHCRR (interregion session recovery), DFHCRSP (CICS interregion communication startup module), DFHZCP (CICS terminal management program), and DFHZCX (which includes DFHZIS2, the interregion service subroutines).

Interregion programs

DFHIRP (interregion communication (SVC) program)

The interregion communication program (DFHIRP) is used to pass data from one region to another in the same processing unit. The programs running in the regions usually are CICS programs, but DFHIRP does not assume that to be the case.

Each user of this program must first issue a LOGON request specifying an 8-character name. This user identifier is added to a table maintained in key 0 storage.

After the user has logged on, CONNECT requests may be issued to establish data paths to other users who have also logged on. The users must cooperate in this process by specifying, when they log on, to whom and from whom they are to be connected and by how many data paths.

After a connection has been established, either end of the connection may issue a SWITCH request to send data to the other end of the connection. The receiver of the data must provide a buffer into which the data is to be written. If the buffer is too small, the receiver is notified of the actual data length and, possibly having obtained a larger buffer, may issue a PULL request to retrieve as much data as is required. After the first data has been sent, the link must be used by each end alternately.

A connection may be broken by either end by issuing a DISCONNECT request. When all links have been disconnected, a user may log off.

When MVS cross-memory services are requested (ACCESSMETHOD(XM) in the RDO CONNECTION definition), DFHXMP is used (DFHIRP performs initialization and termination for DFHXMP); otherwise, communication is performed by DFHIRP running as an SVC. In this case, it is invoked by an SVC call to a startup program (DFHCSVC), which calls the required DFHIRP routine.

DFHXMP (MVS cross-memory program)

When the MVS cross-memory services are used for interregion communication, the SWITCH and PULL functions are performed by DFHXMP, which is entered by issuing a **program call** (PC) instruction instead of an SVC. DFHXMP does not need a commonly addressable buffer or service request blocks (SRBs) to effect data transfer between address spaces.

Code in DFHIRP performs the cross-memory initialization and termination functions for DFHXMP as follows:

LOGON: Acquire and initialize the cross-memory resources (authorization index (AX), linkage index (LX), and entry table (ET)), unless this has already been done by a previous logon in this address space.

CONNECT: Update the authority tables (ATs) of both address spaces to allow each one to establish addressability to the other, unless this was done when a previous connection was established between them.

DISCONNECT: If the last cross-memory connection between a pair of address spaces is being removed, update the caller's AT so that the other system is no longer permitted to access the caller's address space.

LOGOFF: Free the cross-memory resources acquired by logon if they are no longer required by the caller's address space.

CICS address space modules

The CICS address space modules control the handling of requests between this address space and other address spaces. They include several MRO management modules such as DFHCRSP (see DFHCRSP (CICS IRC startup module)) and DFHCRNP (see DFHCRNP (connection manager—CSNC transaction)), and several terminal-control modules (see "DFHZCX (CICS terminal control routines)" on page 302).

These modules provide the CICS address space with a DFHTC-level interface to interregion communication (in the same way as DFHZCP provides a DFHTC-level interface to VTAM). This enables other CICS modules (such as DFHISP) to allocate and execute input/output operations on IRC sessions. The IRC sessions are used for all forms of IRC communication, and the macro-level services available for IRC are broadly the same. Thus DFHISP works for both IRC and intersystem communication (ISC) function shipping.

The functions of each module are as follows:

DFHCRSP (CICS IRC startup module)

Execution of this module makes interregion communication possible between this address space and other address spaces. DFHCRSP, which can be invoked either at system initialization or by the master terminal, allocates the cross-region block (CRB), issues a LOGON request to the SVC routine, and attaches the CSNC transaction (connection manager program, DFHCRNP).

DFHCRNP (connection manager—CSNC transaction)

Interregion communication is controlled by the interregion control program, DFHCRNP, which runs as transaction CSNC. This is attached when CICS first logs on to the interregion program, and it remains attached until interregion communication is closed.

The main purpose of CSNC is to perform housekeeping and control on IRC sessions, and to simulate the access method. Its functions include the following:

1. Establish connections to other address spaces (by issuing CONNECT requests)
2. Detect unsolicited input data on connections and attach requested tasks to process such data
3. Disconnect unallocated (**between-bracket**) sessions during QUIESCE
4. Issue DFHKC AVAIL for any secondary sessions which have become available for reallocation, and are in demand

5. Issue PC RETURN when QUIESCE is complete.

CSNC is attached by DFHCRSP (IRC startup), and waits when it is not processing work. It is resumed by the dispatcher when the MRO work queue ECB has been posted, or the delay interval (if set) has expired and there is delayed work to be retried.

Whenever CSNC is posted, it checks first whether it has been invoked because quiescing of the interregion facility is complete.

- If CSNC has no been resumed to complete interregion quiesce processing, it checks each of the following:
  1. If the “delay-queue” is not empty, CSNC attempts to process any work it finds there. (An element is added to the queue whenever a transaction cannot be attached by CSNC. The system could, for example, have been at maximum tasks or short on storage when the previous attempt was made. It is also possible that a remote system tried to start a new conversation before the local system had freed the required session from an earlier conversation.)
  2. If a new conversation has been received:
     – If this is the first conversation on a new connection, and the connecting region is not a batch region, session recovery is performed. This means that if the name of the secondary connecting matches the name of the secondary connected in the previous session, the old session is bound once again.
     – If there is no match, or if a batch region is connecting, the first available session is allocated.
     – CSNC attempts to attach the required transaction, identified in the attach header included in the data stream. It is possible for a request to arrive for this session before the session has been freed from the transaction that last used it. In such a case, the transaction to be attached is added to the delay-queue.
     – The input data stream is built into a TIOA for the session.
  3. If this region is a secondary, and there is no task associated with the connection, and the connection is in quiesce, CSNC disconnects the session.
  4. If this region is a primary, and it has received a “disconnect” request from the connected secondary, CSNC disconnects the session if:
     – There is no associated TCTTE
     – There is no task associated with the link.
- If CSNC has been resumed to complete interregion quiesce processing, it:
  1. Sends message DFHIR3762 to the CSMT log.
  2. Resumes any suspended mirror tasks with a facility address of zero, so they can detach themselves.
  3. Disable immediate and delay queues. Any remaining work on those queues (for example, old retry work which has not been serviced yet) is automatically discarded.
  4. Logs off from the interregion SVC.
  5. Detaches, using a DFHLFM TYPE=RETURN request.

DFHCRR (CICS session recovery module)

Whenever a new connection is established (via a successful CONNECT request), DFHCRNP links to DFHCRR at the secondary end of the connection (that is, at the source of the connection). DFHCRNP sends a data stream down to the other end of the connection (the primary end) which causes DFHCRNP to link to DFHCRR at the primary end. The two DFHCRRs exchange information in order to determine whether either end of the connection was in doubt when the previous use of the connection was terminated, and, if so, whether the two ends were in sync or out of sync. In the case of an in-doubt connection, the sequence numbers are compared, diagnostics are issued, and the session is freed.

DFHCRC (interregion abnormal exit module)

This module contains the ESTAE exit routine corresponding to the ESTAE macro issued by DFHKESIP. It is invoked if the ESTAE exit, DFHKESTX, decides to continue the abend, or if an X22 abend (which can't be handled by DFHKESTX) occurs.

The purpose of the exit is to free links with other subsystems to which connection has been made by the interregion SVC, and to free links with the SVC itself. This is done by issuing to the SVC a CLEAR request (to break links with other subsystems).

DFHZCX (CICS terminal control routines)

DFHZCX is a load module consisting of a set of object modules, including DFHZIS1 (ISC or IRC syncpoint) and DFHZIS2 (IRC internal functions).

DFHZIS2 provides the following routines:

I/O request routine (IORENT)
Provides a WRITE/WAIT/READ interface to interregion connections.

GETDATA routine (GDAENT)
Retrieves input data from an IRC connection and puts it into a TIOA.

RECEIVE routine (RECENT)
Receives unsolicited data (**begin-bracket** in SNA terms) and checks validity.

DISCONNECT routine (DSCENT)
Cleans up this end of a connection, and issues DISCONNECT request to DFHIRP.

OPRENT routine (OPRENT)
Issues an INSRV request to DFHIRP, in order to allow future connections between this subsystem and a specified subsystem.

RECABRT routine (RCAENT)
Is invoked when an ABORT FMH (FMH07) is received (indicating that the connected transaction has abended). The routine issues a message describing the failure.

STOP routine (STPENT)
Is invoked when communication with other address spaces is to be terminated. The routine issues a QUIESCE request to DFHIRP.

LOGOFF routine (LGFENT)
Is invoked when quiesce is complete (and during system termination and abend processing). The routine issues a LOGOFF request to the SVC routine.

DFHZIS1 also contains routines representing terminal control services which are supported by IRC (in common with VTAM). These routines include PREPARE, SPR, COMMIT, and ABORT.

DFHZCP (CICS terminal management program)

DFHZCP is a load module consisting of a set of object modules, including DFHZARQ (application request handler), DFHZISP (intersystem program allocation routines), and DFHZSUP (startup task).

DFHZARQ is used (in common with all other telecommunication access methods) to handle WRITE/WAIT/READ-level requests against IRC connections (sessions). Routine ZARQIRC in DFHZARQ specifically handles IRC requests by performing SNA request header processing and invoking IORENT (see DFHZCX) in order to perform the I/O on the session.

DFHZISP includes routines such as ALLOCATE and FREE.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:

- AP DDxx, for which the trace levels are IS 1 and IS 2.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 29. Intersystem communication (ISC)

CICS intersystem communication (ISC) allows the following:
- CICS-to-CICS communication
- CICS-to-IMS communication
- CICS-to-LUTYPE6.2 terminal or application communication.

These can be execute simultaneously within the same or a different CEC. ISC can use VTAM LU6.1 or LU6.2 (LU6.2 is preferred for CICS operation). For information about these methods of communication, see the *CICS Intercommunication Guide*.

The facilities provided by ISC include:
- Transaction routing
- Distributed transaction processing
- Function shipping
- Asynchronous processing
- Distributed program link
- SAA Communications interface.

For information about the design and operation of intersystem communication, see Chapter 66, “VTAM LU6.2,” on page 479. For descriptions of the facilities provided by ISC, see Chapter 62, “Transaction routing,” on page 441, Chapter 14, “Distributed transaction processing,” on page 109, Chapter 26, “Function shipping,” on page 277, and Chapter 43, “SAA Communications and Resource Recovery interfaces,” on page 347.

Chapter 30. Interval control

Interval control provides various optional task-related functions based on specified intervals of time, or specified time of day.

Design overview

The following services are performed by interval control in response to a specific request from either an application program or another CICS function:

Time of day

The EXEC CICS ASKTIME command retrieves the current time-of-day in either binary or packed decimal format.

Time-dependent task synchronization

Time-dependent task synchronization provides the user with three optional services:

1. The EXEC CICS DELAY command allows a task to temporarily suspend itself for a specified period of time. When the time has elapsed, the task resumes execution.
2. The EXEC CICS POST command allows a task to be notified when the specified interval of time has elapsed or the specified time of day occurs. The task proceeds to execute while the time interval is elapsing.
3. The EXEC CICS CANCEL command allows a task to terminate its own or another task's request for a DELAY, POST or START service.

Automatic time-ordered transaction initiation

Automatic time-ordered transaction initiation provides for the automatic initiation of a transaction at a specified time of day (or after a specified interval of time has elapsed) and for the sending of data that is to be accessed by the transaction. The user can also cancel a pending request for automatic time-ordered transaction initiation.

Optional user exits are provided as follows:

- Before determining what type of request for time services was issued
- Upon expiration of a previously requested time-dependent event
- If a START request names an unknown terminal.

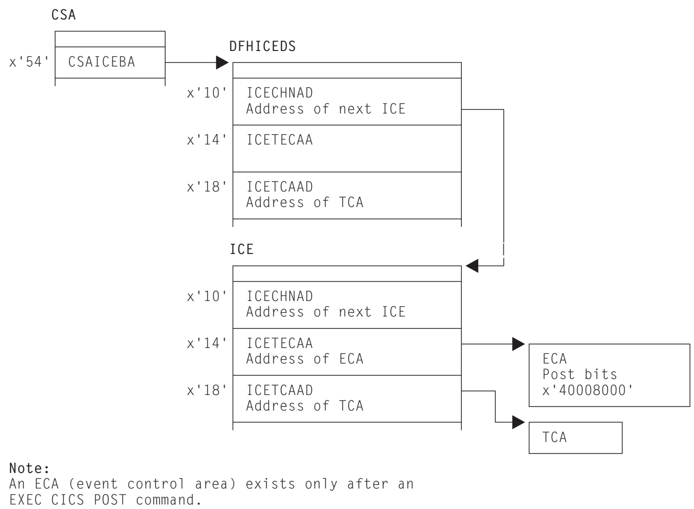
Time-of-day control

The EXEC CICS RESETTIME command causes CICS to reset its internal date and time of day information to accord with that of the operating system. This is done by calling DFHICP with a DFHIC TYPE=RESET macro. This macro is also issued by DFHAPTIM - the program run by the "midnight task" attached by interval control initialization - whenever it is resumed by the TI domain, i.e. at midnight.

DFHICP issues a KETI RESET_LOCAL_TIME call to the TI domain if the reason for the reset was a time of day change. This allows the TI domain to readjust its clocks to the operating system time. DFHICP then calls DFHTAJP to readjust other CICS clocks to match the operating system time and to make any necessary changes to the ICE chain resulting from possible changes in the time-to-expiry of time controlled ICEs. Finally DFHICP scans the ICE chain in order to process any that may have become expired as a result of the time change, and to reset the time interval for which the expiry task, DFHAPTIX, will wait, until the next ICE expires.

Control blocks

An interval control element (ICE—see Figure 66) is created for each time-dependent request received by interval control. These ICEs are chained from the CSA in expiration time-of-day sequence.



Note: An ECA (event control area) exists only after an EXEC CICS POST command.

Figure 66. Interval control element (ICE)

Expired time-ordered requests are processed by Interval Control when called from the DFHAPTIX module, which runs under a system task that has been resumed by the timer domain. The type of service represented by the expired ICE is initiated, if all resources required for the service are available, and the ICE is removed from the chain. If the resources are not available, the ICE remains on the chain and another attempt to initiate the requested service is made later.

See the *CICS Data Areas* manual for a detailed description of this control block.

Modules

DFHAPTIM, DFHAPTIX, DFHICP, DFHICRC, and DFHTAJP

Exits

There are three global user exit points in DFHICP: XICEXP, XICREQ, and XICTENF. See the *CICS Customization Guide* for further information.

Trace

The following point ID is provided for DFHICP:
- AP 00F3, for which the trace level is IC 1.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 31. Language Environment interface

This section describes the run-time interface between CICS and Language Environment®.

Design overview

Communication between CICS and Language Environment is made by calling a special Language Environment interface module (CEECCICS) and passing to it a parameter list (addressed by register 1), which consists of an indication of the function to be performed and a set of address pointers to data values or areas.

Module CEECCICS is distributed in the Language Environment library, but must be copied to an authorized library defined in the STEPLIB concatenation of the CICS startup job stream (see *CICS System Definition Guide*).

All calls to Language Environment are made directly from the CICS language interface module DFHAPLI. This module is called by several components of CICS to perform specific functions. Table 10 lists those functions, and shows the name of the CICS module initiating each function call and the Language Environment call made by DFHAPLI to support the function. The format of each call parameter list is given in "External interfaces" on page 313.

Table 10. Language Environment interface calls

Function	Module	Language Environment call
Terminate Languages	DFHSTP	Partition Termination
Establish Language	DFHPGLK DFHPGLU DFHPGPG	Establish Ownership Type
Start Program	DFHPGLK DFHPGLU	Thread Initialization Run Unit Initialization Run Unit Begin Invocation Run Unit End Invocation Run Unit Termination Thread Termination
Goto	DFHEIP	Perform Goto
Find Program Attributes	DFHEDFX	Determine Working Storage
Initialize Languages	DFHSIJ1	Partition Initialization

The logical relationship between the different calls is shown in Figure 67 on page 310.

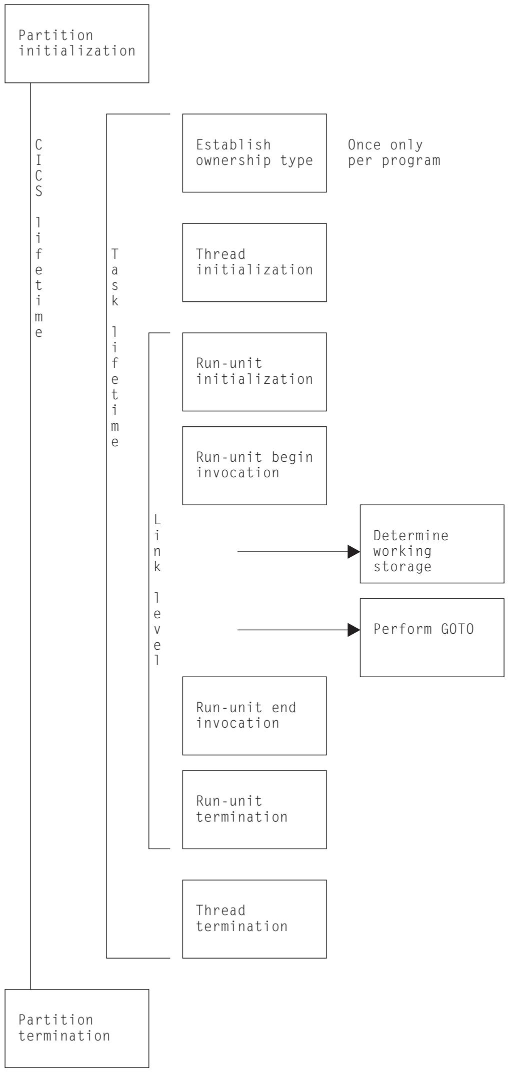


Figure 67. Language Environment interface components

Note: The actual passing of control to CEECCICS is made from the CICS language interface program (DFHAPLI), which provides a single point of contact between CICS and Language Environment. Other modules call DFHAPLI to initiate the desired function.

All calls to DFHAPLI use either the DFHAPLIM macro (for calls from outside the CICS application domain), or the DFHLILIM macro (for calls made from within the application domain).

Establishing the connection

The procedure for establishing the initial connection with Language Environment is as follows:

1. **Load CEECCICS.** At CICS startup, DFHSIJ1 invokes DFHAPLI to “initialize languages”. DFHAPLI issues a BLDL for CEECCICS, followed by an MVS LOAD macro.
2. **Initialize contact with Language Environment.** Contact is first made with Language Environment by having CICS drive the partition initialization function. DFHAPLI attempts partition initialization only if the earlier load of CEECCICS was successful; otherwise, the logic is bypassed.

   If the Language Environment partition initialization is successful, and Language Environment indicates that it can support the running of programs in languages supported by CICS, a flag is set and no further processing takes place.

If the partition initialization function fails, CICS issues error message DFHAP1200.

Application program contact with Language Environment. Whenever a program written in a supported language is run, the application's attempt to make contact with Language Environment fails if the "Language Environment initialization bits" flag is not set. CICS then tries to run the program itself using the basic support for the language. If this fails, CICS then abends the transaction and sets the associated installed resource definition as disabled.

Storage for the transaction

A set of work areas is required during the lifetime of any task that includes one or more programs supported by Language Environment. This set is known as the "language interface work area".

The language interface work area contains storage for the following:
- The largest possible Language Environment interface parameter list (currently 15 parameter elements, but with space allowed for a further three elements)
- A general-purpose register save area for use by DFHAPLI
- A general-purpose register save area for use by Language Environment
- A 240-byte special work area for use by Language Environment as the equivalent of DFHEISTG for CICS
- A 4-byte Language Environment reason code field
- The IOINFO area (see page 316)
- The PGMINFO1 area (see page 316)
- The program termination block (see page 318).

Also, a thread work area is required if Language Environment is involved in the running of the task. The length of a thread work area is a constant value that is notified to CICS by Language Environment during the partition initialization processing. This additional work area is built contiguous with the language interface work area if the transaction is known to contain one or more programs that use Language Environment. When such a program is first encountered, DFHAPLI:
1. Gets from the transaction manager the address of the transaction-related instance data.
2. Flags the data to tell the transaction manager that the transaction runs Language Environment application programs.
3. Adds the length of the language interface work area to the total user storage length for that transaction.

This forces the transaction manager to acquire extra storage, during task initialization, as an extension to the language interface work area. For the first occurrence only, DFHAPLI acquires the thread work area.

Further areas known as run-unit work areas (RUWAs) are required at run time if the transaction includes one or more programs that use Language Environment. The length of an RUWA varies for each program. The lengths required for work areas above and below the 16MB line by Language Environment are notified to CICS during the processing to establish ownership type for that program; thereafter they can be found in the program's installed resource definition. CICS adds to the length for the RUWA above the 16MB line a fixed amount for its own purposes before acquiring the storage.

Storage acquisition

During task initialization, the transaction manager acquires an area of storage, the language interface work area, which is large enough to hold all required data for calls to Language Environment. This area is contiguous with the EXEC interface storage (EIS), and its address is saved in TCACEEPT in the TCA.

The thread work area is usually contiguous with the language interface work area. Its address is always held in CEE_TWA in the language interface work area.

For every link level entered during the execution of the application, a run-unit work area must be acquired by CICS and its address passed to Language Environment during run-unit initialization. Its address is placed in EIORUSTG in the EXEC interface storage (EIS).

Control blocks

The main control block is the language interface work area. It is addressed by TCACEEPT in the TCA. For programs supported by Language Environment, the work area is mapped by the Language_Interface_Workarea DSECT.

Modules

The Language Environment interface is accessed in the language interface program (DFHAPLI) in response to calls from the following modules:

`DFHSIJ1`, `DFHEIP`, `DFHEDFX`, and `DFHSTP`.

Exits

No global user exit points are provided for this interface.

Trace

Trace entries are made on entry to and exit from DFHAPLI.

Point IDs AP 1940 to AP 1945, with a trace level of PC 1, correspond to these trace entries.

The function information is always interpreted.

For entry trace records, the program name and link level are also interpreted where applicable.

For exit trace records, the returned reason code is interpreted.

Also, all calls into and out of the language environments are traced at level 1. The point IDs are: AP1948 to AP 1952.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

(The *CICS Trace Entries* includes tables that show, for entry and exit trace records, the ERTLI function together with any other data items traced.)

The ERTLI function named in the DFHAPLI entry trace is the function requested on the call, while that named in the DFHAPLI exit trace is the ERTLI function most recently processed. There are some situations in which a trace record made on entry to DFHAPLI is not matched by a corresponding exit trace for the same ERTLI function. In particular, after making a call to Language Environment for thread initialization, DFHAPLI does ***not*** return to the caller, but proceeds with “run-unit initialization” and “run-unit begin invocation” before finally returning. Another example is the successful execution of a “perform GOTO” function, which results in DFHAPLI not returning to the caller.

Note: ERTLI refers to the Extended Run-Time Language Interface. This is an extension of the Run-Time Language Interface (RTLI) protocols that were defined to assist communication between CICS and both VS COBOL II and C/370. ERTLI includes communication between CICS and Language Environment.

External interfaces

This section describes the parameter lists and work areas used for the functions provided by the Language Environment interface.

Language Environment interface parameter lists

The following tables show the layout and contents of the parameter lists for the functions provided by the Language Environment interface module CEECCICS.

Table 11. Language Environment PARTITION_INITIALIZATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"10" (= Partition initialization)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token	Yes	8
6	EIBLEN	Length of CICS EIB		F'word
7	TWALEN	Thread work area length	Yes	F'word
8	CELLEVEL	Language Environment-CICS interface level	Yes	F'word
9	GETCAA	Get-CAA routine address		4
10	SETCAA	Set-CAA routine address		4
11	LANGDEF	Language modules defined		32
12	LANGBITS	Language availability bits	Yes	F'word

Table 12. Language Environment ESTABLISH_OWNERSHIP_TYPE parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"50" (= Establish ownership type)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	reserved			
7	reserved			
8	PGMINFO1	CICS-Language Environment program information		48
9	PGMINFO2	Language Environment-CICS program information	Yes	20

Table 13. Language Environment THREAD_INITIALIZATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"20" (= Thread initialization)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token	Yes	8
7	PREATWA	Address of preallocated thread work area		4
8	PGMINFO1	CICS-Language Environment program information		48
9	PGMINFO2	Language Environment-CICS program information		20

Table 14. Language Environment RUNUNIT_INITIALIZATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"30" (= Run-unit initialization)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token		8
7	RTOKEN	Run-unit token	Yes	8
8	PGMINFO1	CICS-Language Environment program information		48
9	PGMINFO2	Language Environment-CICS program information		20

Table 15. Language Environment RUNUNIT_BEGIN_INVOCATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"32" (= Run-unit begin invocation)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token		8
7	RTOKEN	Run-unit token		8
8	PGMINFO1	CICS-Language Environment program information		48
9	PGMINFO2	Language Environment-CICS program information		20
10	IOINFO	Input/output queue details		18
11	RSA	RSA at last EXEC CICS command		F'word

Table 16. Language Environment DETERMINE_WORKING_STORAGE parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"60" (= Determine working storage)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token		8
7	RTOKEN	Run-unit token		8
8	LANG	Program language bits		F'word
9	PGMRSA	Register save area address		4
10	WSA	Working storage address	Yes	4
11	WSL	Working storage length	Yes	F'word
12	SSA	Static storage address (reserved)	Yes	4
13	SSL	Static storage length (reserved)	Yes	F'word
14	EP	Program entry point	Yes	4

Table 17. Language Environment PERFORM_GOTO parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"70" (= Perform GOTO)		F'word
2	RSNCODE	Reason code	Yes	F'word

Table 17. Language Environment PERFORM_GOTO parameter list (continued)

No.	Parameter name	Description	Receiver field	Data length
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token		8
7	RTOKEN	Run-unit token		8
8	LANG	Program language bits		F'word
9	LABEL	Label argument at Handle		F'word
10	RSA	RSA at last EXEC CICS command		F'word
11	CALLERR	Cross call error flag	Yes	F'word
12	ABCODE	Address of TACB abend code		F'word
13	R13	Register 13 value at abend		F'word

Table 18. Language Environment RUNUNIT_END_INVOCATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"33" (= Run-unit end invocation)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token		8
7	RTOKEN	Run-unit token		8
8	PGMINFO1	CICS-Language Environment program information		48
9	PGMINFO2	Language Environment-CICS program information		20
10	PTB	Program termination block		64
11	RSA	RSA at last EXEC CICS command		F'word

Table 19. Language Environment RUNUNIT_TERMINATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"31" (= Run-unit termination)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token		8
7	RTOKEN	Run-unit token	Yes	8

Table 20. Language Environment THREAD_TERMINATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"21" (= Thread termination)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token	Yes	8

Table 21. Language Environment PARTITION_TERMINATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"11" (= Partition termination)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8

Work areas

The following sections describe the work areas required during the lifetime of any task that includes one or more programs that use the Language Environment interface.

IOINFO

The IOINFO area, which is built by DFHAPLI in the CICS-Language Environment work area, is passed to Language Environment on a RUNUNIT_BEGIN_INVOCATION call.

CICS applications cannot use the SYSIN and SYSPRINT data streams because such usage would conflict with the way CICS handles I/O. However, an application may require a general input or output data stream in some situations, for example, where it is necessary to output a message to a program and the program has not been written to expect such output under normal operation.

Three such data streams are architected for this purpose: input, output (normal), and error output. The destinations must be either spools or queues. CICS uses queues, so the file type is always set to "Q". Table 22 shows the transient data queue names that are passed to Language Environment.

Table 22. Transient data queues for use by Language Environment

File type	Language Environment queue name
Input	CESI
Output	CESO
Error output	CESE

Each data stream is identified by a 6-byte control block, and the three control blocks are concatenated to form the IOINFO area, which CICS passes to Language Environment.

IOINFO has this format (in assembler-language code):

```
IOINFO   DS    0CL18    Input/output queue details

STD_IN   DS    0CL6     Standard input file
QORS_IN  DS    CL1      ..file type - "Q" = transient data
TDQ_IN   DS    CL4      ..queue name
SPO_IN   DS    CL1      ..spool class - not used

STD_OUT  DS    0CL6     Standard output file
QORS_OUT DS    CL1      ..file type - "Q" = transient data
TDQ_OUT  DS    CL4      ..queue name
SPO_OUT  DS    CL1      ..spool class - not used

STD_ERR  DS    0CL6     Standard error output file
QORS_ERR DS    CL1      ..file type - "Q" = transient data
TDQ_ERR  DS    CL4      ..queue name
SPO_ERR  DS    CL1      ..spool class - not used
```

PGMINFO1

The PGMINFO1 area, which is built by DFHAPLI in the CICS-Language Environment work area, is passed to Language Environment during these interface calls:

ESTABLISH_OWNERSHIP_TYPE
THREAD_INITIALIZATION
RUNUNIT_INITIALIZATION
RUNUNIT_BEGIN_INVOCATION
RUNUNIT_END_INVOCATION

When both CICS and Language Environment are capable of supporting it, the separate calls to Language Environment for Rununit Initialisation and Rununit Begin Invocation are combined into a single call. This single call is a Rununit Initialisation call with additional parameters indicating

1. make the combined call
2. whether CICS believes the RUWA being passed has already been passed to Language Environment, and so need not be completely initialised by LE.

PGMINFO1 has this format (in assembler-language code):

```
PGMINFO1      DS    0F
P1_LENGTH     DS    F         Length of PGMINFO1
RULANG        DS    XL4       Language as defined by user
ASSEMBLER     EQU   X'80'     ..Assembler
C             EQU   X'40'     ..C
COBOL         EQU   X'20'     ..COBOL
PLI           EQU   X'10'     ..PL/I
LE370         EQU   X'04'     ..Language Environment

RULOADMOD     DS    0F
RULOADA       DS    A         Run-unit load module address
RULOADL       DS    F         Run-unit load module length

ENTRY_STATIC  DS    0F
RUENTRY       DS    A         Run-unit entry point address
RUSTATIC      DS    A         Modified entry address
RWA_31        DS    A         Address of run-unit storage
                                above 16MB
RWA_24        DS    A         Address of run-unit storage
                                below 16MB
APAL          DS    A         Application argument list
                                address
RTOPTS        DS    A         Run-time options
RTOPTSL       DS    F         Length of run-time options
RUNAMEP       DS    A         Pointer to the program name
PGMINFO1L     EQU   *-PGMINFO1
```

PGMINFO2

The PGMINFO2 area, which forms part of the PPT entry for the running program, is filled in by Language Environment on successful completion of an ESTABLISH_OWNERSHIP_TYPE call; and is subsequently passed by CICS to Language Environment during these interface calls:
THREAD_INITIALIZATION
RUNUNIT INITIALIZATION
RUNUNIT_BEGIN_INVOCATION
RUNUNIT_END_INVOCATION

PGMINFO2 has this format (in assembler-language code):

```
PGMINFO2   DS    0F
PRGINLEN   DS    FL4       Length of PGMINFO2 extension
PLBRWA31   DS    F         Length of 31-bit RUWA
PLBRWAA    EQU   X'80'     ..31-bit storage required (C/370)
PLBRWAL    DS    FL3       ..Length of 31-bit RUWA
PLBRWA24   DS    F         Length of 24-bit RUWA

PLBLANG    DS    0CL4      Language availability byte
PLBLANG1   DS    X
```

```
PLBCEEEN   EQU   X'80'     ..Language Environment
                              enabled
PLBCEELA   EQU   X'40'     ..Language Environment
                              language known
PLBMIXED   EQU   X'20'     ..Mixed/single language
PLBCOMPT   EQU   X'10'     ..Compatibility
PLBEXECU   EQU   X'08'     ..Language Environment
                              executable
PLBASSEM   EQU   X'04'     ..Assembler-language program
PLBC370    EQU   X'02'     ..C/370 program
PLBCOBL2   EQU   X'01'     ..VS COBOL II program
PLBLANG2   DS    X
PLBOSCOB   EQU   X'80'     ..OS/VS COBOL program
PLBPLI     EQU   X'40'     ..OS PLI program
PLBTYPE3   DS    X         Reserved
PLBTYPE4   DS    X         Reserved
PLBMEMID   DS    FL4       Language member ID
PLBED      EQU   *-PGMINFO2
```

Program termination block

The program termination block (PTB), which is built by DFHAPLI in the CICS-Language Environment work area, is passed to Language Environment on a RUNUNIT_END_INVOCATION call.

It has this format (in Assembler-language code):

```
CELINFO    DS    0F
PCHK       DS    0CL32     Abend information
           DS    CL8
PCHK_PSW   DS    CL8       ..PSW
PCHKINTS   DS    0CL8      ..Interrupt data
PCHK_LEN   DS    XL2       ....Instruction length
PCHK_INT   DS    XL2       ....Interrupt code
PCHK_ADR   DS    FL4       ..Exception address
PCHK_GR    DS    AL4       ..A(GP registers at abend)
PCHK_FR    DS    AL4       ..A(FP registers at abend)
PCHK_AR    DS    AL4       ..A(AX registers at abend)
PCHK_EX    DS    AL4       ..A(Registers at the last time
                              a CICS command was issued)
CNTCODE    DS    0CL4      Continuation code
CONT1      EQU   X'40'    ..retry using registers
CONT2      EQU   X'20'    ..retry using PSW
           DS    BL3      Reserved
RTRY       DS    0CL20
RTRY_AD    DS    FL4      ..Retry address
RTRY_PM    DS    AL4      ..A(Program mask)
RTRY_GR    DS    AL4      ..A(GP registers)
RTRY_FR    DS    AL4      ..A(FP registers)
RTRY_AR    DS    AL4      ..A(AX registers)
```

Chapter 32. Master terminal program

The master terminal program enables dynamic control of the system. Using this function an operator can change the values of parameters used by CICS, alter the status of system resources, terminate tasks, and shut down the CICS system.

Design overview

The master terminal program is invoked by the CEMT transaction. The user enters a command to INQUIRE about or SET the status of a set of resources, and the command outputs a display that shows the resultant status of the resources. For a CEMT SET command, this display can be overtyped to alter the status of most of the resources displayed.

Commands are analyzed using the same techniques as the command interpreter described in Chapter 9, “Command interpreter,” on page 87. A language table is used to define the syntax of commands and the contents of parameter lists which must be passed to DFHEIP to allow execution. In effect, each CEMT command results in the execution of a series of EXEC CICS INQUIRE and SET commands.

The master terminal program is also used by the CEST and CEOT transactions, which provide subsets of the functions available with CEMT. CEST is for supervisory operators and allows access to a limited set of resources. CEOT only allows changes to the status of the operator’s own terminal.

Modules

Module	Function
DFHEMTP	Invoked by CEMT. Checks that the terminal is suitable. Obtains and initializes working storage. Loads the language table DFHEITMT. Links to DFHEMTD.
DFHEOTP	Same as DFHEMTP but invoked by CEOT and loads the language table DFHEITOT.
DFHESTP	Same as DFHEMTP but invoked by CEST and loads the language table DFHEITST.
DFHEMTD	Receives data from the terminal and sends back a display. Analyzes commands and overtypes. Constructs parameter lists for DFHEIP, which it calls. Deals with PF keys.
DFHEITMT	Command language table for CEMT.
DFHEITOT	Command language table for CEOT.
DFHEITST	Command language table for CEST.

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

Chapter 33. Message generation program

The message generation program provides an interface for sending CICS messages to the terminal user only.

Design overview

The input to the message generation program (DFHMGP) consists of the binary number of the message to be produced, the identifier of the component issuing the message, and any information to be inserted in the message. DFHMGP builds the complete message using a prototype held in the message prototype control table, also known as the message generation table (DFHMGT). The message text itself is held not in DFHMGT but in the message domain, from which it is retrieved by the DFHMGPME routine (a component of the DFHMGP load module) when required. DFHMGP finally sends the message to the appropriate terminal.

The prototype statements are invocations of the DFHMGM TYPE=TEXT macro, and are contained in copybooks held in DFHMGT.

The message prototype control table consists of a series of copybooks, DFHMGTnn, each of which contains 1 through 100 messages. They are arranged in such a way that each DFHMGTnn copybook contains prototypes for messages that have identifiers of the form DFHccnnxx, where cc is the 2-character identifier of the component issuing the message, nn is the numerical part of the copybook name, and xx is in the range 00 through 99. For example, the prototype for message DFHAC2214 (belonging to the AC component) is in copybook DFHMGT22.

Within each copybook are invocations of DFHMGM in ascending message number order. All messages sent to the terminal end user have both OPTION=NLS and COMPID specified on their DFHMGM invocations.

The main operands of the DFHMGM TYPE=TEXT macro are:
- MSGNO = actual message number
- COMPID = 2-character identifier of component issuing the message (this forms part of the message identifier)
- OPTION = any special options, for example, (NLS) for messages that require NLS enabling.

Other operands are provided on the DFHMGM invocations, but in general these are no longer used.

Modules

DFHMGP, DFHMGT

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for this function:
- AP 00E0, for which the trace level is AP 1.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 34. Message switching

This function provides the user with a general-purpose message-switching capability while CICS is running.

The facility, which can route messages to one or more destinations, is initiated by the CMSG transaction, or a user-chosen replacement, read from the terminal. For further information about this transaction, see the *CICS Supplied Transactions* manual.

Design overview

Message switching runs as a task under CICS. A terminal operator requests activation of this task by entry of the CMSG transaction identifier (or another installation-defined 4-character transaction identifier), followed by appropriate parameters. After it has been initiated, message switching interfaces with CICS basic mapping support (BMS) and CICS control functions.

Although message switching appears conversational to the terminal operator, the message switching task is terminated with each terminal response. Conversation is forced, if continuation is possible, by effectively terminating the transaction with an EXEC CICS RETURN TRANSID(xxxx) command, where xxxx is the transaction identifier taken from the task's PCT entry.

Figure 68 shows the message-switching interfaces.

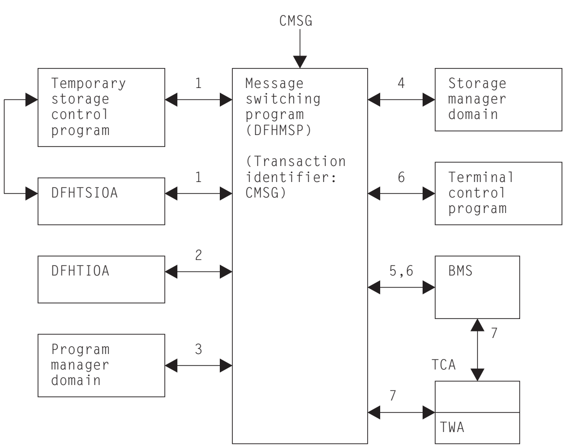


Figure 68. Message-switching interfaces

Notes:

1. If the first 4 characters of the terminal input/output area (TIOA) (not including a possible set buffer address (SBA) sequence from an IBM 3270 Information Display System) do not match the transaction identifier in the task's PCT entry, this task must have started as part of a conversation in which a previous task has set up the next transaction identifier. A "C" immediately following the transaction identifier is also a forced continuation. In such a case, information has been stored in, and has to be retrieved from, temporary storage (using a record key of 1-byte X'FC', 4-byte terminal identifier, and 3-byte C"MSG") to allow the task to resume where it left off.
2. The operands in the input TIOA are processed and their values and status are stored in the TWA.
3. If a ROUTE operand specifies terminal list tables (TLTs) for a standard routing list, the program manager domain is called to load the requested TLTs.
4. Message switching requests storage areas for:

- Building route lists (one or more segments, each of which has room for the number of destinations specified by MSRTELNG, an EQU within the program).
- Constructing a record to be placed in temporary storage.
- Providing the message text to BMS in any of the following situations:
  - Message parts from previous inputs exceed the current TIOA size
  - A message is completed in the current TIOA but has parts from previous inputs
  - A heading has been requested but the message in the current TIOA is too close to TIOADBA to allow the header to be inserted.

5. Message switching requests BMS routing functions by means of the DFHBMS TYPE=ROUTE macro. The message text is sent using DFHBMS TYPE=TEXTBLD, and completion of the message is indicated by DFHBMS TYPE=PAGEOUT. BMS returns the status of destinations and any error indications in response to the DFHBMS TYPE=CHECK macro.
6. Message switching interfaces with BMS using DFHBMS TYPE=(EDIT,OUT) and with CICS terminal control using DFHTC TYPE=WRITE for the IBM 3270 Information Display System only, in providing responses to terminals. These can indicate normal completion, signal that input is to continue, or provide notification of input error.
7. Like any other task, message switching has a task control area (TCA) in which values may be placed prior to issuing CICS macros, and from which any returned values can be retrieved after an operation. All values for the DFHBMS TYPE=ROUTE macro are placed in the TCA because they are created at execution time. The TWA is used for storing status information (partly saved in temporary storage across conversations) and space for work. The DFHMSP module is reentrant.

Control blocks

See the list of control blocks in Chapter 5, “Basic mapping support,” on page 29.

Modules

DFHMSP (the message switching program) is invoked by the CMSG transaction. DFHMSP’s purpose is to route a message entered at the terminal to one or more operator-defined terminals or to other operators.

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

External interfaces

See Figure 68 on page 323 for external calls made to other areas or domains.

Chapter 35. Multiregion operation (MRO)

CICS multiregion operation (MRO) enables CICS regions that are running in the same MVS image, or in the same MVS sysplex, to communicate with each other. MRO does not support communication between a CICS system and a non-CICS system such as IMS.[3]

ACF/VTAM and SNA networking facilities are not required for MRO. The support within CICS that enables region-to-region communication is called **interregion communication**

The facilities provided by MRO include:
- Transaction routing
- Distributed transaction processing
- Function shipping
- Asynchronous processing
- Distributed program link.

For more information about the design and implementation of interregion communication facilities, see Chapter 28, "Interregion communication (IRC)," on page 293. For descriptions of the facilities provided by MRO, see:
- Chapter 13, "Distributed program link," on page 107
- Chapter 14, "Distributed transaction processing," on page 109
- Chapter 26, "Function shipping," on page 277
- Chapter 62, "Transaction routing," on page 441.

3. The external CICS interface (EXCI) uses a specialized form of MRO link to support: communication between MVS batch programs and CICS; DCE remote procedure calls to CICS programs.

Chapter 36. Node abnormal condition program

DFHZNAC is a CICS program used by terminal control to analyze abnormal terminal conditions that are logical unit or node errors detected by VTAM. VTAM notifies the CICS terminal control program that there is a terminal error, and the terminal control program places the terminal out of service. The terminal control program then invokes DFHZNAC, which writes any error messages to the CSNE transient data destination.

Design overview

The node abnormal condition program (DFHZNAC) can be called for any of several reasons:

- As a central point of control for most VTAM-related error situations, error actions can be standardized in table form, allowing for easy addition and alteration to the way conditions are processed.
- Some exception conditions that are not errors are also processed by DFHZNAC, but some exception conditions that are errors are not processed by DFHZNAC.
- It provides a single point of user interface to those who want to change the default actions for an error requiring at most one user program (NEP)—see Chapter 37, “Node error program,” on page 331.

To process conditions that are not associated with a known terminal, the dummy TCTTE is used. It is invoked by placing a TCTTE on the system error queue with a 1-byte code relating to the condition. Placing it on the queue makes the TCTTE ‘temporary OUTSERV’ (TCTTESOS); that is, the decision is pending the outcome of DFHZNAC.

The activate scan routine (DFHZACT) is responsible for attaching the CSNE transaction to run DFHZNAC; this is done during CICS initialization. The CSNE transaction remains in the system until CICS or VTAM is quiesced. If DFHZNAC itself abends, or VTAM is closed and then restarted, DFHZACT attaches a new CSNE transaction when there is more work for DFHZNAC to do.

There is only ever one CSNE transaction in the system at any one time. (This should not be confused with the CSNE transaction that is attached by the remote delete processing of autoinstall.)

Once DFHZNAC has been called, it runs down the system error queue, processing each error for each TCTTE on the queue. When there is no more work to be done, DFHZNAC suspends itself, to be resumed by DFHZACT when further processing is required.

Note that the system error queue need not be empty before DFHZNAC terminates; errors can be left on the queue to be processed later. For example, in an XRF environment, some error codes cannot be handled until the alternate CICS system has taken over; that is, it has passed the ‘initialization complete’ stage. If DFHZNAC is passed a TCTTE indicating such an error, it leaves that entry on the queue.

Node abnormal condition program (NACP) processing involves mapping the error code (placed into the TCTTE by a DFHZERRM macro call) to a set of actions, performing any specific processing for that error code, accumulating the actions for all the error codes in that TCTTE, and then performing the actions.

Figure 69 on page 328 shows the NACP error code processing. The numbers in Figure 69 refer to the following notes, which use the table entry for DFHZC3424 as the example:

```
DFHZNCM MSGNO=3424,
        E1=S88,
        E2=NULL,
        E3=NULL,
        E4=NULL,
        ACT=(ABSEND,ABRECV,ABTASK,CLSDST,SIMLOG),
        CODE=NSP02,
        TYPE=ENTRY
```

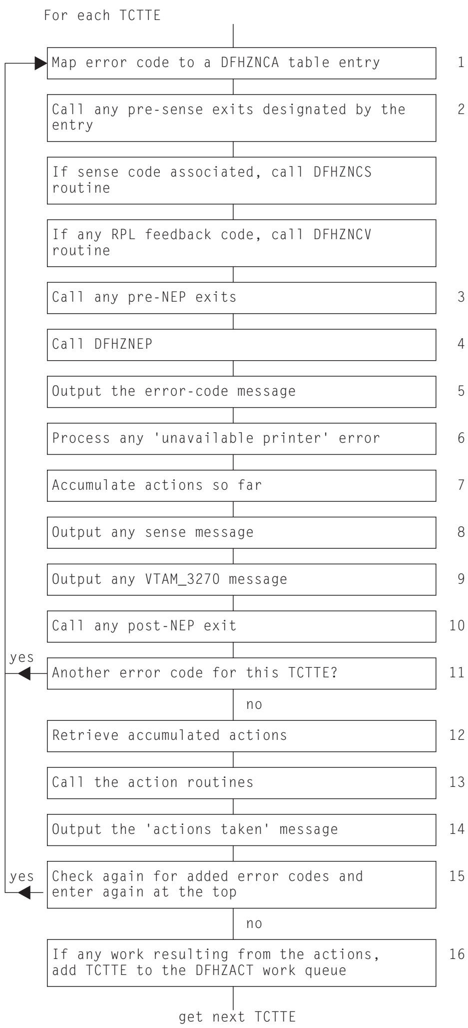


Figure 69. NACP error code processing

Notes:

1. The error codes in TCTEVRC* and default actions are defined in the VTAM-associated errors section of the *CICS Trace Entries*.

   In the example, TCTVRC5 contains X'5C', which equates to TCZNSP02 (ref CODE=NSP02).
2. Errors that involve SNA sense have it saved in TCTEVNSS. It is processed by code in copy book DFHZNCS.
3. Call any pre-NEP exits specified by the table entry; for example, E1=S88 references routine NAPES88.
4. Call the node error program (NEP), passing a parameter list via a COMMAREA. This call may or may not change the default actions. The operation of the NEP is described in the *CICS Customization Guide* and the Chapter 37, “Node error program,” on page 331.
5. Output error-code message associated with the table entry (DFHZC3424 from MSGNO=3424) to the CSNE log.
6. Check for ‘unavailable printer error’—this caters for a screen copy request that is unable to find an eligible printer if the first choice is unavailable.
7. Because there can be multiple error codes, the actions are accumulated now and performed together later.

8. Output any sense message resulting from the DFHZNCS call, to the CSNE log.
9. Output any VTAM_3270 message resulting from the DFHZNCS call (if it was non-SNA) to the CSNE log.
10. Call the post-NEP exit, if any (E4=NULL, no routine).
11. Loop for each error code in TCTEVRC*.
12. When all the error codes for this TCTTE that can be processed at this time have been processed, retrieve the actions that have been accumulated, such as ACT=(ABSEND, ABRECV, ABTASK, CLSDST, SIMLOG).
13. Call the action routine to process each of the actions.
14. Output the 'actions taken' message DFHZC3437 to the CSNE log.
15. Check again for any error codes added asynchronously while the CSNE transaction was running.
16. Queue any work resulting from the actions to the activate scan routine.

Control blocks

DFHZNAC references CSA, its own TCA, JCA, TCT prefix, TIOA, NIB, PCT, SIT, TCTWE, VTAM RPL, VTAM ACB, and the NACP/NEP communication area.

As would be expected, however, the processing mainly concerns access to the TCTTE, and to the NACP/NEP communication area (COMMAREA), which is mapped by the DFHNEPCA DSECT.

See the *CICS Data Areas* manual or the *CICS Customization Guide* for a detailed description of the NEP communication area.

Modules

Module	Function
DFHZNAC	Processes the system error queue of TCTTEs and contains the central structure of NACP, outlined in Figure 69 on page 328. It contains the following copy books:
DFHZNCA	This copy book contains the exit routines for each error code and the error code table itself built by DFHZNCM macros.
DFHZNCE	Links to the user node error program (DFHZNEP) and responds to the action flag settings in the NACP/NEP COMMAREA.
DFHZNCS	Processes the SNA sense codes and contains the sense code tables built using a combination of DFHZMJM and DFHZNCM macros.
DFHZNCV	Contains the VTAM return code table.
DFHZNCM	The macro to build the error code table.
DFHZMJM	The macro to build the sense code table.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for the node abnormal condition program, as part of terminal control:

- AP FCxx, for which the trace levels are TC 1, TC 2, and Exc
- AP FD7E, for which the trace level is TC 1.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Statistics

The only statistical field that DFHZNAC updates is TCTTETE. Because DFHZNAC is the main module for terminal errors, it has primary responsibility for updating the node error count.

Chapter 37. Node error program

CICS provides a user-replaceable node error program, DFHZNEP, which assists the user in the following ways:

- It provides a general environment within which it is easy for users to add their own error processors.
- It provides the fundamental error recovery actions for a VTAM 3270 network.
- It serves as the default node error program (NEP), where the user selects a NEP at system initialization.

The DFHZNEP program can be one of the following:

- The CICS-supplied default NEP
- A skeleton sample NEP generated using the DFHSNEP macro
- A user-written NEP generated using the DFHSNEP macro.

Design overview

The purpose of the NEP is to allow user-dependent processing whenever a communication system event is reported to CICS. An example of the processing that can be done is to analyze the event and override the default action set by DFHZNAC. When NEP processing is complete, control returns to DFHZNAC.

The default node error program sets the ‘print TCTTE’ action flag (TWAOTCTE in the user option byte TWAOPT1, defined in DFHNEPCA) if a VTAM storage problem has been detected; otherwise, it performs no processing, and leaves the action flags set by DFHZNAC unchanged.

The skeleton sample NEP provided by CICS can provide extended error handling for VTAM terminals, and is generated by means of the DFHSNEP macro. This procedure is described in the *CICS Customization Guide*.

The DFHSNEP macro can also be used to generate a user-written NEP. Interactions between the applications and VTAM depend on characteristics of the transactions and the installation. Each system has different characteristics. The CICS-provided skeleton NEP is a framework for a user-written NEP to handle network error conditions that may be unique to a particular installation.

Guidance information about NEP coding is given in the *CICS Recovery and Restart Guide*. Reference information about NEP coding is given in the *CICS Customization Guide*.

Modules

DFHZNEP

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided specifically for this function; however, trace entries are made from DFHZNAC immediately before and after calling the node error program.

Point IDs AP FC71 and AP FC72, with a trace level of TC 1, correspond to these trace entries.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 38. Program control

The program control program, DFHPCP, is an interface routine which supports DFHPC LINK, ABEND, SETXIT and RESETXIT calls issued in other CICS modules and invokes the appropriate program manager domain function.

In previous releases DFHPCP provided the functions that are now provided by the Program Manager Domain, and other domains.

Design overview

Services in response to requests

The following services are performed by DFHPCP in response to DFHPC requests from other CICS functions, where those functions have not been converted to use domain interfaces :

Link (LINK)
Builds a parameter list and issues DFHPGLK FUNCTION(LINK) domain call.

Handle Abend (SETXIT)
If SETXIT macro specifies an abend routine address, then DFHPCP builds a parameter list and issues a DFHPGHM FUNCTION(SET_ABEND) OPERATION(HANDLE) call. If SETXIT macro does not specify an abend routine address, then DFHPCP builds a parameter list and issues a DFHPGHM FUNCTION(SET_ABEND) OPERATION(CANCEL) call.

RESETXIT
DFHPCP builds a parameter list and issues a DFHPGHM FUNCTION(SET_ABEND) OPERATION(RESET) call. If SETXIT macro does not specify an abend routine address, then DFHPCP builds a parameter list and issues a DFHPGHM CANCEL call.

Abend (ABEND)
If it is an ABEND request without an existing TACB, then the parameter list is built for this abend. A DFHABAB(CREATE_ABEND_RECORD) is issued to build the TACB. Else a DFHABAB(UPDATE_ABEND_RECORD) is issued with the name of the failing program is issued. A DFHABAB(START_ABEND) call is then made to issue the abend. If the DFHABAB(START_ABEND) call returns control to this module, it is because the exit XPCTA has been invoked and modified the return address. Control is passed to the modified address in the requested execution key.

Modules

DFHEPC

Call mechanism

Branched to from DFHEIP.

Entry address

DFHEPCNA. Stored in the CSA in a field named CSAEPC.

Purpose

DFHEPC is DFHEIP's program control interface. It supports the following EXEC CICS requests

- LINK
- XCTL
- RETURN
- LOAD

- RELEASE
- ABEND
- HANDLE ABEND

It routes a local request to the PG domain, or to DFHABAB (EXEC CICS ABEND) It routes a remote EXEC CICS LINK request to the intersystem module, DFHISP.

Called by

DFHEPC is called exclusively by DFHEIP.

Inputs

The application parameter list.

Outputs

Updated EIB.

Operation

LINK
: If SYSID is remote, ships the link request through the DFHISP module.

  If SYSID is local:
  - Builds parameter list and calls DFHPGLE FUNCTION(LINK_EXEC)
  - Checks the response.
  - If response indicates the program is remote, ships the link request through the DFHISP module.
  - Sets up EIBRESP (and, if needed, EIBRESP2).
  - Returns control to DFHEIP.

XCTL
: Builds parameter list and calls DFHPGXE FUNCTION(PREPARE_XCTL_EXEC)

  Checks the response

  Sets up EIBRESP (and, if needed, EIBRESP2).

  If the PGXE request failed, then returns control to DFHEIP

  If the PGXE request was successful, then return control to DFHAPLI as for EXEC CICS RETURN. (DFHAPLI will then invoke the program specified on EXEC CICS XCTL).

RETURN
: Builds parameter list and calls DFHPGRE FUNCTION(PREPARE_RETURN_EXEC) (this call is bypassed if there are no options (COMMAREA, TRANSID, INPUTMSG) specified on EXEC CICS RETURN

  . Checks the response

  . Sets up EIBRESP (and, if needed, EIBRESP2).

  . If the PGRE request failed, then returns control to DFHEIP

  . If the PGRE request was successful (or was bypassed), then return control to DFHAPLI which completes the return to the calling program or to Transaction Manager.

LOAD
: Builds parameter list and calls DFHPGLD FUNCTION(LOAD_EXEC)

  Checks the response

  Sets up EIBRESP (and, if needed, EIBRESP2).

  If the PGLD request was successful, then set the return parameters in the application parameter list.

  Returns control to DFHEIP.

RELEASE
: Builds parameter list and calls DFHPGLD FUNCTION(RELEASE_EXEC)

Checks the response

Sets up EIBRESP (and, if needed, EIBRESP2).

Returns control to DFHEIP.

HANDLE ABEND
For HANDLE ABEND PROGRAM, perform resource security check and check whether program name is known.

Builds parameter list and calls DFHPGHM FUNCTION(SET_ABEND)
- OPERATION(HANDLE) for HANDLE ABEND PROGRAM or LABEL
- OPERATION(CANCEL) for HANDLE ABEND CANCEL
- OPERATION(RESET) for HANDLE ABEND

Checks the response

Sets up EIBRESP (and, if needed, EIBRESP2).

Returns control to DFHEIP.

ABEND
Builds parameter list and calls DFHABAB FUNCTION(CREATE_ABEND_RECORD) and FUNCTION(START_ABEND).

DFHABAB START_ABEND does not normally return, as control is passed to a program or label specified on a HANDLE ABEND, or the program is terminated abnormally.

The XPCTA user exit can request retry. In this case DFHABAB START_ABEND returns to DFHEPC passing back the retry parameters. DFHEPC sets the registers and other values and branches to the specified retry address.

How loaded

At CICS startup, as part of the building of the CICS nucleus. The nucleus is built by DFHSIB1, which uses its nucleus build list to determine the content and characteristics of the CICS nucleus.

Exits

There are two global user exit points in DFHEPC: XPCREQ and XPCREQC.

There are two global user exit points in DFHABAB: XPCABND and XPCTA.

There are two global user exit points in DFHAPLI1: XPCFTCH and XPCHAIR.

There is one global user exit point in DFHERM: XPCHAIR.

There is one global user exit point in DFHUEH: XPCHAIR.

For further information, see the *CICS Customization Guide*.

Trace

The following point IDs are provided for entry to and exit from DFHPCPG:
- AP 2000, for which the trace level is PC 1
- AP 2001, for which the trace level is PC 1.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 39. Program error program

CICS provides a dummy program error program (DFHPEP) that does nothing except give control back to the abnormal condition program (DFHACP), which is invoked during transaction abend processing.

You can provide some additional routines to handle programming errors. For instance, it is possible to disable the transaction code associated with the program in error, thus preventing the recurrence of the error until it can be corrected; send messages to the end-user terminal; initiate a new transaction; or record abend information in transient data.

Design overview

To provide corrective action in response to a programming error, you can code a program error program (DFHPEP). This program can then be assembled and link-edited to replace the dummy DFHPEP.

If provided, this program is invoked by the abnormal condition program (DFHACP) whenever a task terminates due to a task abnormal condition. However, it will **NOT** be called if a task is terminated due to an attach failure (for example the transaction is not defined) or when CICS deliberately terminates a task to alleviate a stall.

The user can perform any kind of corrective action within a program error program.

Guidance information about PEP coding is given in the *CICS Recovery and Restart Guide*. Reference information about PEP coding is given in the *CICS Customization Guide*.

Control blocks

The control block associated with the program error program is: DFHPEP_COMMAREA, the commarea passed to DFHPEP.

See the *CICS Data Areas* manual for a detailed description of this control block.

Modules

DFHPEP

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

Chapter 40. Program preparation utilities

The program preparation utilities consist of the command-language translators, which are utility programs that run offline to translate CICS application programs using command-level CICS requests. They convert the EXEC commands into call statements in the language in which the EXEC commands are embedded. Versions of the translator program are available for:
- COBOL (DFHECP1$)
- PL/I (DFHEPP1$)
- C (DFHEDP1$)
- Assembler language (DFHEAP1$).

Design overview

The command-language translators manage storage by creating a stack from a single area allocated at the start of the program.

Because the input is free-format, the translators move it into a buffer area that can hold data spanning two or more source records. The analysis of the source is mainly table driven.

The translators build the replacement source code for each EXEC command in a form appropriate to the language:
- For COBOL, the replacement code contains a series of MOVE statements, followed by a CALL statement.
- For PL/I, the replacement code contains a declaration of an entry variable followed by a CALL statement. These statements are contained within a DO group.
- For C, the replacement code contains a function call (dfhexec) and may also contain assignment statements.
- For assembler language, the replacement code is an invocation of the DFHECALL macro.

Errors in the source can be detected. Spelling corrections are made to the source, and any unrecognizable or duplicate keywords and options are ignored. For COBOL, PL/I, and C, the translator produces error diagnostics that are collected together on the output listing. The assembler language translator, however, produces error diagnostics in the translated output following the EXEC command in which the error occurred.

Modules

DFHECP1$, DFHEPP1$, DFHEDP1$, DFHEAP1$

Exits

Global user exit points are not applicable to offline utilities.

Trace

Trace points are not applicable to offline utilities.

Chapter 41. Remote DL/I

An overall description of DL/I database support is given in Chapter 15, “DL/I database support,” on page 119. This section gives information that is specific to remote DL/I.

Design overview

This section outlines what you must do to define remote DL/I support, and describes the functions of remote DL/I.

System definition

For a CICS system that supports only remote databases you must, in addition to providing the usual definitions that are required for function shipping, code a PSB directory (PDIR) using the DFHDLPSB macro. Every PDIR entry must have SYSIDNT specified. The PDIR system initialization parameter must be coded specifying the suffix of the PDIR.

DL/I PSB scheduling

When a CICS task requests the scheduling of a DL/I PSB by means of an EXEC DLI SCHEDULE request or DL/I PCB call, and the request is for a remote PSB, control is passed to DFHDLIRP. DFHDLIRP allocates a remote scheduling block (RSB) and issues a DFHIS TYPE=CONVERSE macro to ship the scheduling request to the owning system.

Database calls

For a remote DL/I database call, a DFHIS TYPE=CONVERSE macro is issued to ship the request to the owning system. The return codes are passed back to the user in the user interface block (UIB).

DL/I PSB termination

If a remote PSB is terminated, the actions performed are:

1. Free the RSB and local program communication block (PCB) storage.
2. If the DL/I PSB termination was not caused by a CICS syncpoint, request one now.

Control blocks

Figure 70 illustrates some of the control blocks used to support remote DL/I.

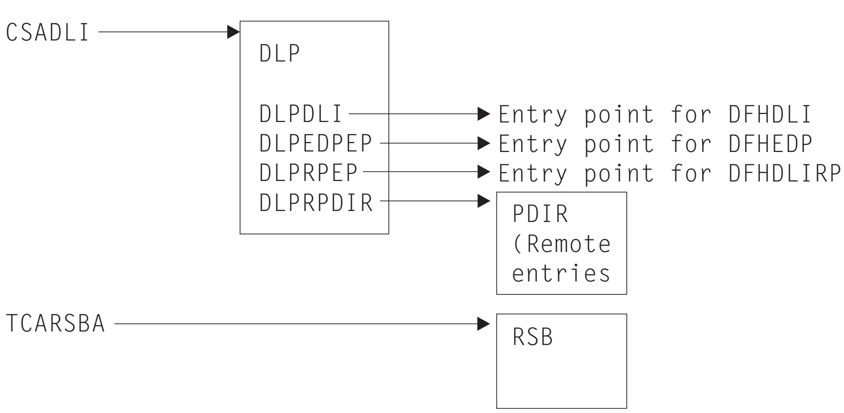


Figure 70. Some control blocks used for remote DL/I support

The DL/I interface parameter list (DLP) is described in “DL/I interface parameter list (DLP)” on page 121.

Remote DL/I

The remote PSB directory (PDIR) contains an entry for each remote PSB that can be used from an application program.

The remote scheduling block (RSB) is acquired when a CICS task issues a PSB schedule request for a remote PSB. The RSB is freed when the task issues a SYNCPOINT or a DLI TERM request.

See *CICS Data Areas* for a detailed description of these control blocks.

Chapter 42. Resource definition online (RDO)

The CEDA transaction creates and alters the definitions of system resources in the CICS system definition (CSD) data set.

RDO provides:

- Online transactions that can be used to **inspect**, **change**, and **install** resource definitions:
  - CEDA (inspect, change, and install)
  - CEDB (inspect and change)
  - CEDC (inspect only).
- A programmable interface to the CEDA transaction, using an EXEC CICS LINK command in the application program to invoke DFHEDAP directly. (For further information, see the *CICS Customization Guide*.)
- A set of system programmer API command (the EXEC CICS CREATE commands) for creating CICS resources dynamically.
- An offline utility, DFHCSDUP, to inspect or change resource definitions. (For a description of this utility, see Chapter 10, "CSD utility program (DFHCSDUP)," on page 89.)

Design overview

Resource definitions are maintained on the CICS system definition (CSD) data set. The resource definitions in the CSD data set can be viewed and changed using either the online CEDx transactions, or the offline utility DFHCSDUP.

Installation of resource definitions makes the definitions available to the running CICS system. Resource definitions can be installed at these times:

- When CICS is cold started, using the GRPLIST system initialization parameter.
- During a run of CICS, using the CEDA transaction.

When resource definitions are installed, they are made available through the appropriate resource managers.

Modules

The relationships between the components of RDO, and the components of some of the services it uses, are shown in Figure 71 on page 344.

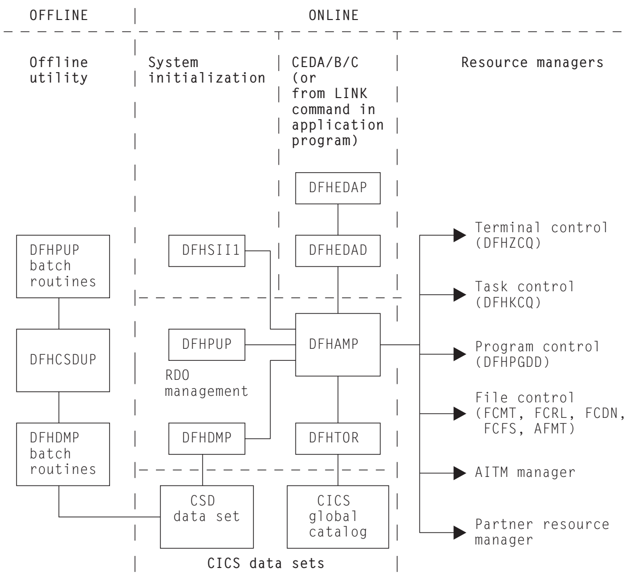


Figure 71. RDO interfaces

DFHEDAP and **DFHEDAD** control the CEDA, CEDB, and CEDC transactions. They provide screen management for the transactions, and invoke DFHAMP to implement any actions that are required.

DFHSII1 invokes DFHAMP when CICS is cold started, to install resource definitions for the current run. These resource definitions are specified by the GRPLIST system initialization parameter. DFHSII1 passes the GRPLIST system initialization parameter to DFHAMP.

DFHAMP, the allocation management program, manages all requests to view, change, and install resources. It uses the services provided by other parts of RDO, and by the resource managers:

- DFHAMP invokes **DFHPUP** and **DFHDMP** to read, write, and update resource definitions on the CSD data set:
  – DFHPUP, the parameter utility program, converts resource definition data between the parameter list format provided by DFHAMP and the record format needed by the CSD.
  – DFHDMP, the CSD management program, manages I/O of resource definition data to and from the CSD data set.
- DFHAMP invokes **DFHTOR**, the terminal object resolution program, to merge TERMINAL, TYPETERM, CONNECTION, and SESSION definitions:
  – When requests are made to install TERMINALs, TYPETERMs, CONNECTIONs, and SESSIONs, DFHTOR merges TYPETERM and TERMINAL information, and also CONNECTION and SESSION information, and passes this merged information back to DFHAMP.
  – DFHAMP passes the merged definitions to DFHZCQ to install in the running CICS system. Any merged TERMINAL definitions that are to be used as autoinstall terminal models are passed to the autoinstall terminal model (AITM) manager.
  – When TYPETERM definitions are installed, DFHTOR records the information about the CICS global catalog for subsequent use.
  – When the CHECK command is issued, DFHTOR checks the appropriate TERMINAL, TYPETERM, CONNECTION, and SESSION definitions for consistency.
- DFHAMP calls the appropriate resource managers to install resources in the running CICS system:
  – **DFHZCQ** is invoked to install CONNECTION, SESSION, and TERMINAL definitions.
  – **DFHAMXM** is invoked to install TRANSACTION and PROFILE definitions.
  – **DFHPGDD** is invoked to install PROGRAM, MAPSET, and PARTITIONSET definitions.

– These subroutine “gates” are called to install resources related to file control:
  FCMT, for FCT entries
  FCRL, for LSR pools
  FCDN, for DSN blocks
  FCFS, to open and close files
  AFMT, for AFCT entries for files.
– The **AITM manager** is invoked, using an AITM ADD_REPL_TERM_MODEL subroutine call (see Chapter 4, “Autoinstall terminal model manager,” on page 23), to install autoinstall terminal models.
– The **partner resource manager** is invoked, using a PRPT ADD_REPLACE_PARTNER subroutine call (see Chapter 97, “Partner resource manager,” on page 1045), to install partner resources for the SAA communications interface.

DFHEICRE processes all the EXEC CICS CREATE commands. It builds an internal DEFINE command for the resource to be created, and passes it to DFHCAP for interpretation. The encoded command is then passed directly to DFHAMP to install the resource in the running system. The CSD file is not accessed at all during this processing.

DFHCSDUP, the offline CICS system definition utility program, uses batch versions of routines from DFHPUP and DFHDMP to read, write, and update resource definitions on the CSD data set (see Chapter 10, “CSD utility program (DFHCSDUP),” on page 89).

For a detailed description of how the CEDA transaction handles terminal resources, see Chapter 56, “Terminal control,” on page 405.

Exits

The XRSINDI global user exit is invoked at each install or EXEC CICS CREATE.

Trace

The following point IDs are provided, with a trace level of AP 1:
- AP 00EB (DFHAMP)
- AP 00EC (DFHDMP)
- AP 00EF (DFHTOR)
- AP 00E2 (DFHPUP).

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 43. SAA Communications and Resource Recovery interfaces

This section describes the CICS implementation of the Communications and Resource Recovery elements of the Systems Application Architecture® Common Programming Interface (also known as the SAA Communications and SAA Resource Recovery interfaces respectively).

The SAA Communications and Resource Recovery interfaces are both call-based application programming interfaces that are common across all programming languages and across hardware systems.

The common programming interface (CPI) component of CICS, also sometimes known as the CP component, provides application programming interfaces that conform to SAA specifications for Communications and Resource Recovery interfaces.

Note: This CICS component does **not** currently handle any other SAA interface elements.

The CPI component is part of the AP domain, and is shipped as object code only (OCO).

The **SAA Communications interface** allows CICS applications to communicate via APPC (LU6.2) links to partner applications on any system that conforms to SAA standards. This interface consists of a set of defined verbs as program calls that are adapted for the language being used. For further information about the general call-based API, see the *SAA CPI Communications Reference* manual, SC26-4399.

The SAA Communications interface in CICS provides an alternative to the existing application interface for distributed transaction processing (see page 109). A single transaction can use EXEC CICS commands for one conversation while using SAA Communications calls for another (separate) conversation. Also, one end of a conversation can use EXEC CICS commands while the other end uses SAA Communications calls. However, it is not possible to use a mixture of EXEC CICS commands and SAA Communications calls on the same end of a conversation.

The **SAA Resource Recovery interface** provides an SAA application programming interface for commit and backout of recoverable resources. This interface consists of two defined verbs as program calls that are adapted for the language being used:

SRRCMIT
: Commit

SRRBACK
: Backout

For further information, see the *SAA CPI Resource Recovery Reference* manual, SC31-6821.

The SAA Resource Recovery interface in CICS provides an alternative to the use of EXEC CICS SYNCPOINT and EXEC CICS SYNCPOINT ROLLBACK commands. The SRRCMIT call is equivalent to the EXEC CICS SYNCPOINT command, and the SRRBACK call is equivalent to the EXEC CICS SYNCPOINT ROLLBACK command. A single application can use SAA Resource Recovery calls, EXEC CICS commands, or a mixture of both.

Design overview

This section describes the SAA Communications and Resource Recovery interfaces.

The SAA Communications interface

When an application issues an SAA Communications call, control passes via the DFHCPLC application link-edit stub to the common programming interface program (DFHCPI), which in turn passes the request to the DFHCPIC program load module. DFHCPIC verifies the parameters, checks the conversation state, and (if required) issues a DFHLUC macro call to invoke the LU6.2 application request logic module (DFHZARL). For details of DFHZARL, see Chapter 14, “Distributed transaction processing,” on page 109.

Figure 72 shows how the SAA Communications interface support relates to CICS intersystem communication (ISC) using VTAM LU6.2. The numbers in Figure 72 refer to the notes that follow it. CMxxxx represents a program call defined in the SAA Communications interface.

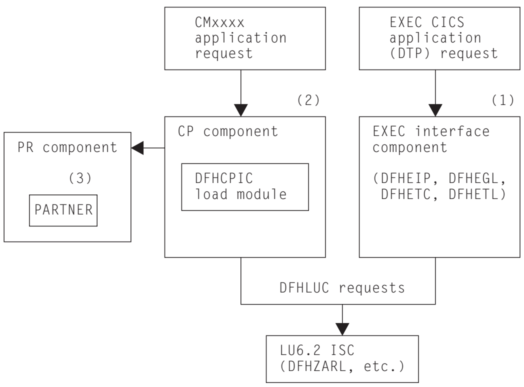


Figure 72. SAA Communications application request processing

Notes:

1. Distributed transaction processing (DTP) allows a transaction using EXEC CICS commands to communicate with a transaction running in another system. This is carried out by DFHEIP and related EXEC interface processor modules. For a VTAM LU6.2 intersystem link, each request is converted into DFHLUC macro requests that call DFHZARL.
2. The SAA Communications interface is implemented by the DFHCPIC load module within the CP (or CPI) component. DFHCPIC maps the CMxxxx application requests into DFHLUC macro calls.
3. To begin a conversation, the SAA Communications interface requires specific information (side information) about the partner program, including its name and system details. This is implemented within CICS as an RDO object called the PARTNER, which is encapsulated by the partner resource manager (PR) component. Further details of the PR component are given under Chapter 97, “Partner resource manager,” on page 1045.

Using the SAA Communications interface on recoverable conversations

When using the SAA Communications interface on recoverable conversations (that is, conversations with the synclevel set to CM_SYNC_POINT), DFHLUC syncpoint requests are routed to DFHZARL via the SAA Communications interface syncpoint request handler (DFHCPSRH) in the DFHCPIC load module. This allows the conversation state to be tracked.

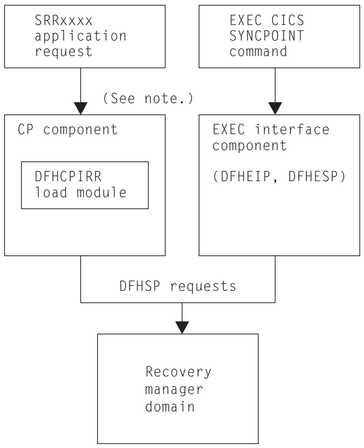
For the equivalent EXEC CICS synclevel 2 conversations, DFHLUC syncpoint requests pass directly to DFHZARL.

The SAA Resource Recovery interface

When an application issues an SAA Resource Recovery call, control passes via the DFHCPLRR application link-edit stub to the common programming interface program (DFHCPI), which in turn passes the request to the DFHCPIRR program load module. DFHCPIRR verifies the parameters, and (if required) issues an appropriate DFHSP macro call: DFHSP TYPE=USER for SRRCMIT, or DFHSP TYPE=ROLLBACK for SRRBACK.

Figure 73 on page 349 shows how the SAA Resource Recovery interface support relates to the processing of EXEC CICS SYNCPOINT commands. The number in the figure refers to the accompanying note.

SRRxxxx represents a program call defined in the SAA Resource Recovery interface, namely, SRRBACK or SRRCMIT.



Note: The SAA Resource Recovery interface is implemented by the DFHCPIRR load module within the CP (or CPI) component. DFHCPIRR maps SRRxxxx application requests into DFHSP macro calls.

Figure 73. SAA Resource Recovery application request processing

Functions provided by the CPI component

Table 23 summarizes the external subroutine interfaces provided by the CPI component. It shows the subroutine call formats, the level-1 trace point IDs of the modules providing the functions for these formats, and the functions provided.

Table 23. CPI component's subroutine interfaces

Format	Trace	Function
CPIN	AP 0C01 AP 0C02	START_INIT COMPLETE_INIT
CPSP	AP 0CD0 AP 0CD1	SYNCPOINT_REQUEST

CPIN format, START_INIT function

The START_INIT function of the CPIN format is used to attach a CICS task to perform initialization of the CPI component.

Input parameters

None.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	GETMAIN_FAILED ADD_SUSPEND_FAILED

CPIN format, COMPLETE_INIT function

The COMPLETE_INIT function of the CPIN format is used to wait for the initialization task attached by the START_INIT function to complete processing.

Input parameters

None.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

`OK|DISASTER|KERNERROR`

[REASON]
is returned when RESPONSE is DISASTER. It has this value:

`INIT_TASK_FAILED`

CPSP format, SYNCPOINT_REQUEST function

The SYNCPOINT_REQUEST function of the CPSP format is used to send LU6.2 syncpoint flows on recoverable conversations using the SAA Communications interface, and to update the conversation state as required.

Input parameters

CPC_ADDRESS
is the address of the SAA Communications conversation control block (CPC).

LUC_ADDRESS
is the address of the DFHLUC parameter list.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have either of these values:

`OK|KERNERROR`

Modules

Module	Function
DFHAPTRF	Trace interpreter for the SAA Communications and Resource Recovery interfaces
DFHCPARH	SAA Communications application request handler (entry processor for all application calls to the DFHCPIC load module, routing them to the appropriate DFHCPCxx module)
DFHCPCxx	Components of the DFHCPIC load module, each object module typically handling a different CMxxxx application request
DFHCPDUF	Offline system dump formatter for CP keyword
DFHCPI	Common programming interface program (link-edited with DFHEIP and DFHAICBP to form the DFHAIP load module)
DFHCPIN1	Initialization management program for the SAA Communications and Resource Recovery interfaces
DFHCPIN2	Runs as a CICS task to perform initialization for the SAA Communications and Resource Recovery interfaces
DFHCPIR	SAA Resource Recovery entry processor, handling all calls to the DFHCPIRR load module
DFHCPLC	Link-edit stub for applications using the SAA Communications interface
DFHCPLRR	Link-edit stub for applications using the SAA Resource Recovery interface
DFHCPSRH	SAA Communications syncpoint request handler (part of the DFHCPIC load module)

Exits

No global user exit points are provided for this component.

Trace

The following point ID is provided for this component:
- AP 0Cxx, for which the trace levels are CP 1, CP 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 44. Statistics utility program (DFHSTUP)

This chapter provides a general overview of the collection of CICS statistics as well as describing the operation of the offline statistics utility program (DFHSTUP). For more information about using the DFHSTUP utility program, see the *CICS Operations and Utilities Guide*.

An operator interface to all online statistics functions is provided by the CEMT transaction. The equivalent programmable interface is provided by the EXEC API.

Statistics may be collected at user-specified intervals from the startup to the shutdown of a CICS system. Statistics may also be requested, resulting in the collection of data for the period between the last time statistics were reset and the time the request was made.

Statistics are also collected at system quiesce or logical end of day; this data is written to the SMF data set as for a normal interval collection.

An option is provided by the statistics domain to allow the user to specify whether interval statistics are to be collected. The statistics domain calls each domain in turn to reset the statistics fields at every interval when statistics are collected. Statistics (particularly interval statistics) can be used for capacity planning and performance tuning. For further information about this aspect, see the *CICS Performance Guide*.

There is a great similarity between CICS statistics data and CICS performance class monitoring data. Statistics data is collected on a resource basis, whereas performance class monitoring collects similar data on a transaction basis. Statistics can therefore be viewed as resource monitoring.

Design overview

CICS statistics support is divided into the following components:

1. The operator interface. This component is responsible for interfacing to the various CICS-supported terminals, analyzing the input string and then invoking the statistics domain to perform the appropriate management operation. This function is provided by the CEMT transaction, and also by the EXEC API.
2. The statistics domain. This component is responsible for managing statistics interfaces, for example, SMF and EXEC API.
3. The statistics update logic. This code is inline in the relevant resource manager. In this way the control function of statistics is centralized, but the management and updating of the statistics fields is given to the resource owner.
4. The statistics data collection and reset. For all collection types except unsolicited (see below), the collection mechanism is the same. The owning domain is invoked by statistics domain to supply a record that contains the domain's statistics. When this record has been formed, the domain then calls statistics domain to place the data on the SMF data set.

   There are five types of collections:

   a. Interval. The collection interval default is 3 hours. This may be changed by the user. The minimum value is 1 minute, the maximum 24 hours. On an interval collection, each called domain collects and resets its statistics counters. No action is taken if the statistics recording status is OFF.
   b. Requested. Statistics may be requested using the PERFORM STATISTICS function provided by the CEMT transaction or the EXEC API. The data recorded is for the period between the last time statistics were reset and the time the request was made. Statistics are reset on an interval, end-of-day, or requested-reset collection; they can also be reset, without a collection, when changing the statistics recording status from ON to OFF, or from OFF to ON.

      This type of collection can obtain statistics from some or all domains, as requested. Each called domain collects, but does not reset, its statistics counters.

      Requested statistics are collected even if the statistics recording status is OFF.

c. Requested-reset. This collection is similar to requested statistics, except that it always obtains statistics for all domains, and each called domain resets its statistics counters after collection.

   Requested-reset statistics are collected even if the statistics recording status is OFF.

d. End-of-day. This collection occurs when the system is quiescing. A logical end-of-day time may be specified. The default time is midnight. This is primarily for continually running systems. The collection is then made at this time, and the called domain collects and resets its statistics counters.

   End-of-day statistics are collected even if the statistics recording status is OFF.

   Daily systems that are taken down after midnight should change the logical end of day to a time when the system is not operational.

   If the user wishes to simulate shutdown statistics, the interval can be set to 24 hours. An end-of-day report, which contains total figures for the CICS run up to the end of the day, can then be printed by DFHSTUP.

e. Unsolicited. For dynamically allocated and deallocated resources, the resource records its statistics just before it is deleted; for example, an autoinstall terminal that logs off and is thereby deleted. USS statistics are written to SMF regardless of the statistics recording status (STATRCD).

By default DFHSTUP formats the statistics for all types of collection, for all the specified APPLIDs. However, if you specify the EXTRACT control parameter but not COLLECTION TYPE, only the extract exit is invoked and no other statistics output is produced.

5. The statistics formatting control. The offline utility DFHSTUP opens the statistics data set, which is an unloaded SMF data set, and the I/O interfaces to that data set. This routine then browses the data set and formats the statistics.

   Reports may be produced for any or all of the five types of statistics collections.

   DFHSTUP also provides the option of producing a summary report for selected CICS applids. The summary report is constructed from all the statistics contained in the interval, requested-reset, end-of-day, and unsolicited collections. Requested statistics are not involved in the production of the summary report.

6. The extract statistics reporting function. This is a DFHSTUP exit that takes statistics data from the input SMF data set and passes it to a user program for processing in order to create tailored statistics reports. DFH0STXR is a sample program designed to exploit the extract reporting function. There are also two skeleton exits; an assembler extract exit called DFH£STXA, and a COBOL extract exit called DFH0STXC. These show the format and structure of the interface between DFHSTUP and the extract exit.

   Specifying the extract statistics reporting function changes the default DFHSTUP report settings. If you specify only the EXTRACT control statement, only the extract exit is driven; other DFHSTUP reports are suppressed. If EXTRACT is specified, other statistics report control statements, such as SUMMARY, must also be specified to ensure that the appropriate reports are produced.

DFHSTUP operation

DFHSTUP runs as a separate MVS job and extracts all or selected entries from the unloaded SMF data set. The types of entries to be processed by this program are specified in the SYSIN data set. Entries that can be selected for processing include:

- All entries—the default
- Entries written for specified applids
- Entries written for specified resource types
- Entries written for specified collection types, that is, interval, requested, requested-reset, end-of-day, or unsolicited
- Entries written during a specified period of time.

You can also select:

- The page size; the default is 60 lines per page.

- Whether output is to be printed in mixed case or all uppercase; the default is to print in mixed case.
- The summary report option; by default, it is not selected.

Further information about using DFHSTUP is given in the *CICS Operations and Utilities Guide*.

Modules

Module	Function
DFH£STXA	Skeleton assembler extract exit
DFH0STXC	Skeleton COBOL extract exit
DFH0STXR	DFHSTUP extract sample program
DFHST03X	VTAM statistics summary formatter
DFHST04X	Autoinstall terminals statistics summary formatter
DFHST06X	Terminal statistics summary formatter
DFHST08X	LSRPOOL resource statistics summary formatter
DFHST09X	LSRPOOL file statistics summary formatter
DFHST14X	ISC/IRC statistics summary formatter
DFHST16X	Table manager statistics summary formatter
DFHST17X	File control statistics summary formatter
DFHST21X	ISC/IRC attach-time statistics summary formatter
DFHST22X	FEPI statistics summary formatter
DFHSTD2X	CICS DB2 statistics summary formatter
DFHSTDBX	DBCTL statistics summary formatter
DFHSTDSX	Dispatcher domain statistics summary formatter
DFHSTDUX	Dump domain statistics summary formatter
DFHSTE15	DFSORT interface to E15 user exit
DFHSTE35	DFSORT interface to E35 user exit
DFHSTEJX	Enterprise Java domain statistics summary formatter
DFHSTIIX	IIOP domain statistics summary formatter
DFHSTIN	DFSORT E15 user exit input routine
DFHSTLDX	Loader domain statistics summary formatter
DFHSTLGX	Log manager domain summary statistics formatter
DFHSTMNX	Monitoring domain statistics summary formatter
DFHSTOT	DFSORT E35 user exit output routine
DFHSTPGX	Program manager domain statistics summary formatter
DFHSTRD	Read interface subroutines
DFHSTRMX	Recovery manager domain statistics summary formatter
DFHSTSJX	JVM domain statistics summary formatter
DFHSTSMX	Storage manager domain statistics summary formatter
DFHSTSOX	Sockets domain statistics summary formatter
DFHSTSTX	Statistics domain statistics summary formatter
DFHSTTQX	Transient data statistics summary formatter
DFHSTTSX	Temporary storage domain statistics summary formatter
DFHSTU03	VTAM statistics formatter
DFHSTU04	Autoinstall terminals statistics formatter
DFHSTU06	Terminal statistics formatter
DFHSTU08	LSRPOOL resource statistics formatter
DFHSTU09	LSRPOOL file statistics formatter
DFHSTU14	ISC/IRC statistics formatter
DFHSTU16	Table manager statistics formatter
DFHSTU17	File control statistics formatter
DFHSTU21	ISC/IRC attach-time statistics formatter
DFHSTU22	FEPI statistics formatter
DFHSTUD2	CICS DB2 statistics formatter

Statistics utility program (DFHSTUP)

Module	Function
DFHSTUDB	DBCTL statistics formatter
DFHSTUDS	Dispatcher domain statistics formatter
DFHSTUDU	Dump domain statistics formatter
DFHSTUEJ	Enterprise Java domain statistics formatter
DFHSTUII	IIOP domain statistics formatter
DFHSTULD	Loader domain statistics formatter
DFHSTULG	Log manager domain statistics formatter
DFHSTUMN	Monitoring domain statistics formatter
DFHSTUP1	PRE_INITIALIZE
DFHSTUPG	Program manager domain statistics formatter
DFHSTURM	Recovery manager domain statistics formatter
DFHSTURS	User domain statistics formatter
DFHSTURX	User domain statistics summary formatter
DFHSTUSJ	JVM domain statistics formatter
DFHSTUSM	Storage manager domain statistics formatter
DFHSTUSO	Sockets domain statistics formatter
DFHSTUTQ	Transient data statistics formatter
DFHSTUST	Statistics domain statistics formatter
DFHSTUTS	Temporary storage domain statistics formatter
DFHSTUXM	Transaction manager domain statistics formatter
DFHSTWR	Write interface subroutines
DFHSTXMX	Transaction manager domain statistics summary formatter

Chapter 45. Storage control macro-compatibility interface

DFHSMSCP is responsible for handling all requests for storage services that are made by using the routine addressed by CSASCNAC in the CICS common system area (CSA). DFHSMSCP is called by some parts of the CICS AP domain containing DFHSC macros.

DFHSMSCP converts all requests into calls to the storage manager domain, and its main function is to get or free storage.

Design overview

The input to DFHSMSCP, set up by the macro used for the invocation, or directly by the calling program, consists of the following TCA fields:

- TCASCTR—the storage control request byte. This can contain one of the following values:
  - X'80' GETMAIN, in conjunction with:
    - X'40' Initialize storage
    - X'20' Conditional

      Storage class in bits 3 through 7 (the resulting SMMC GETMAIN storage class name is given in parentheses where this differs from the first name):
      - X'00' 1WD, treated as SHARED
      - X'04' LINE
      - X'05' TERMINAL or TERM
      - X'0C' USER (becomes CICS24)
      - X'0D' TRANSDATA or TD
      - X'13' SHARED (becomes SHARED_CICS24)
      - X'14' CONTROL
  - X'40' FREEMAIN, in conjunction with:
    - X'01' TCTTE address supplied.
- TCASCIB—the 1-byte value to which storage is to be initialized.
- TCASCNB—the 2-byte field giving the number of bytes requested on the GETMAIN.
- TCASCSA—the 4-byte address of the storage that was obtained or the storage to be freed.

Modules

DFHSMSCP

Exits

No global user exit points are provided for this function.

Trace

The point IDs for this function are of the form AP F1xx; the corresponding trace levels are SC 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 46. Subsystem interface

The subsystem interface is a mechanism by which the MVS operating system communicates with its underlying subsystems at certain critical points in its processing.

CICS is required to be defined as a formal MVS subsystem for the following purposes:
- Multiregion operation (MRO)
- Shared database support
- Console message handling.

Functional overview

An MVS subsystem consists of two control blocks and a set of functional routines, all resident in common memory. The control blocks are:

SSCT The subsystem communication table, which contains the 4-character name of the subsystem and a pointer to the SSVT.

SSVT The subsystem vector table, which contains a list of the subsystem function codes that the subsystem supports, and the addresses of the functional routines that support them.

The subsystem is **active** when the SSCT contains a nonzero pointer to the SSVT, and **inactive** when the pointer is zero.

Subsystem definition

Each subsystem is defined to MVS by an entry in an IEFSSNxx member of SYS1.PARMLIB. (See the *OS/390 MVS Initialization and Tuning Guide*, SC28-1751.) Each subsystem can be defined with an initialization routine and some initialization parameters. The CICS subsystem is defined with an initialization routine of DFHSSIN, and an initialization parameter that specifies the name of an additional member of SYS1.PARMLIB, which contains further CICS-specific subsystem parameters. These parameters specify whether the console message handling facility is required.

Design overview

When the recommended initialization routine DFHSSIN is specified, the CICS subsystem is initialized during the master scheduler initialization phase of the MVS IPL. The CICS-specific subsystem parameters are read from SYS1.PARMLIB, and the subsystem vector table is created. The supporting subsystem function routines are loaded into common memory and their addresses are stored into the subsystem vector table. If everything is successful, the CICS subsystem is made active by storing the address of the subsystem vector table in the subsystem communication table.

Console message handling

At startup, a CICS region that supports console message handling notifies the CICS subsystem of its existence, by using the CICS SVC to issue a subsystem interface call for the ‘generic connect’ function with a CONNECT subfunction. The subsystem notes the creation of the new region and, if this is the first such CICS region to become active, invokes a service of MVS console support called “subsystem console message broadcasting”. The message broadcasting service causes all subsequent console messages to be broadcast to all subsystems that have expressed an interest in receiving them, including the CICS subsystem. This MVS service can also be activated by other products, for example, NetView®.

If the message broadcasting service has been activated, either by CICS or by another product, the CICS subsystem examines ***all*** messages issued by WTO macros in any address space, but it intercepts and modifies only the following:
- Messages beginning with “DFH” that are issued under any CICS TCB, including those CICS regions that do not have console message handling support.

These messages are reformatted to contain the CICS applid for the region in a standard position in the message.

Because the CICS subsystem receives control after JES has recorded a console message in the job's message log, messages in the job log do not appear to be reformatted. The messages are only reformatted on the operator consoles and in the MVS system log.

If the original message is a long one, inserting the CICS applid can cause the message to exceed the maximum length for an MVS console message. In this case, the original message is suppressed (that is, does not appear on the console), and the reformatted message is issued using the MVS multiple-line console message service to split the message text over several lines. Both the original message and perhaps several instances of the reformatted multiple-line message appear in the job log, but only one copy of the reformatted message is displayed on the console.

- Messages that redisplay, on operator consoles or in the MVS system log, MODIFY commands that are directed towards CICS and contain signon passwords for the CESN transaction.

  These messages are reformatted with the passwords replaced by asterisks, so that the original passwords are not exposed.

As each TCB terminates, it issues an 'end of task' subsystem call, which is broadcast to all active subsystems. Likewise, as each address space terminates, it issues an 'end of memory' subsystem call, which is also broadcast to all active subsystems. When it receives either of these calls, the CICS subsystem first calls the end-of-memory routine in DFHIRP; then, if the terminating address space is known by the subsystem, it invokes the 'generic connect' function with a DISCONNECT subfunction.

The DISCONNECT subfunction notes the termination of the CICS address space and, if this is the last CICS containing console message handling support to terminate, notifies the "subsystem console message broadcasting" support that the CICS subsystem is no longer interested in receiving broadcast console messages. Nevertheless, if another product has kept console message broadcasting active, the CICS subsystem continues to reformat messages from CICS regions that do not have console message handling support.

Control Blocks

DSECT	Function
DFHSABDS	The CICS subsystem anchor block (SAB). This is used to contain global subsystem-related information that is common to all CICS regions in the MVS image. It is used to record the options specified in the DFHSSInn member of SYS1.PARMLIB. It contains a pointer to a bit map that records which MVS address spaces contain an active CICS. It also contains the address of the subsystem control table extension (SCTE) used by IRC, and the address of the CEC status tracking information used by XRF.
IEFJSCVT	The subsystem communication table (SSCT). This is an MVS control block. There is one SSCT for each subsystem, including the primary job entry subsystem (JES) as well as CICS.
IEFJSSVT	The subsystem vector table (SSVT). This is an MVS control block. There is one SSVT for each active subsystem. It contains a lookup table for determining which function codes are supported by the subsystem, and a list of the entry points for all the supporting function routines.

Figure 74 on page 361 shows these control blocks.

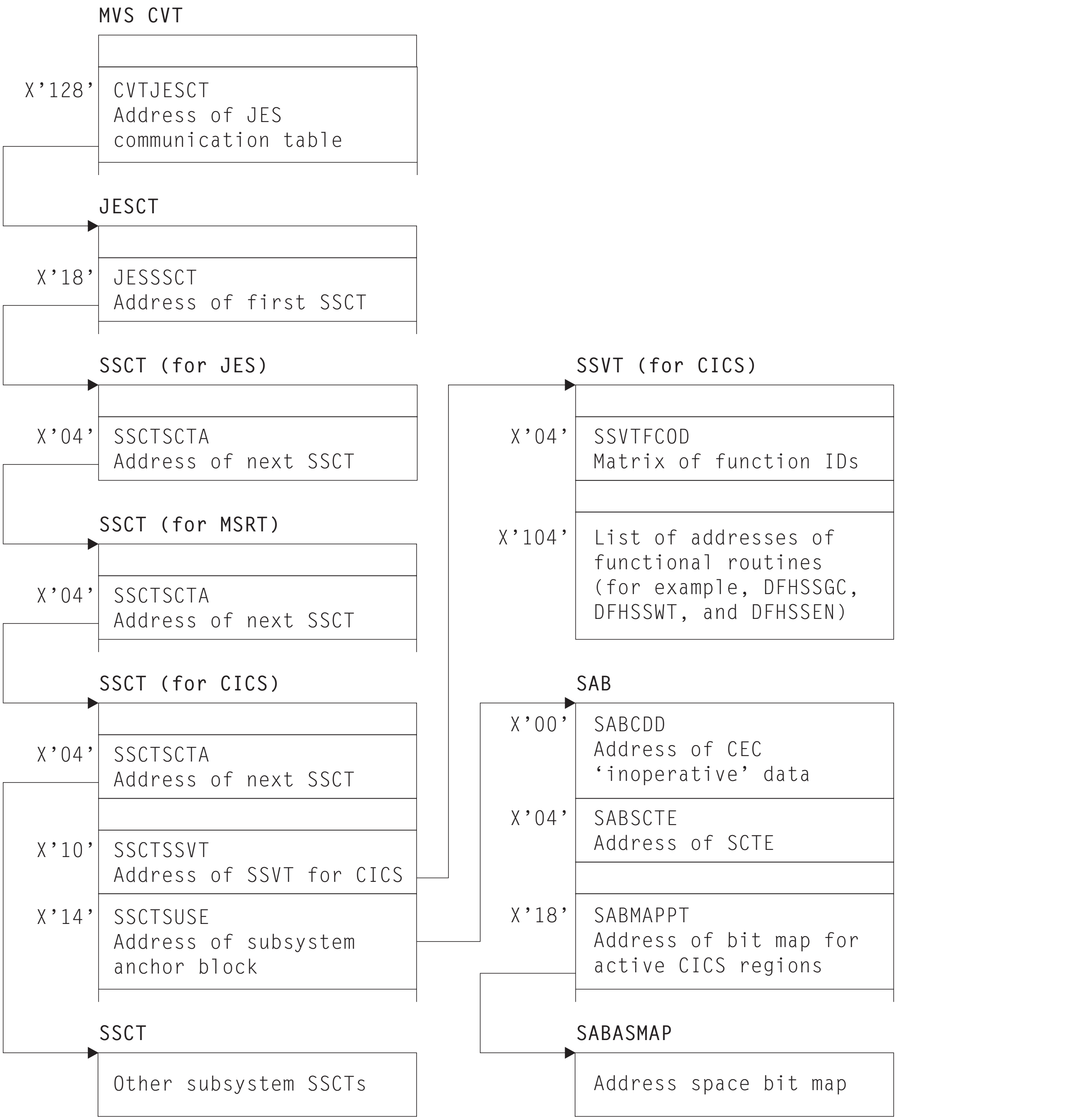


Figure 74. Control blocks associated with the subsystem interface

Modules

Module	Function
DFHSSIN	Subsystem initialization routine for the CICS subsystem. Reads in subsystem parameters from member DFHSSInn of SYS1.PARMLIB, creates SSVT, loads function modules into MVS common storage.
DFHSSEN	End-of-task and end-of-memory functional module. Calls DFHIRP's EOT/EOM routine. Issues 'generic connect' if terminating region or job-step task is in the CICS address space map.

Module	Function
DFHSSGC	The generic connect functional module. CONNECT subfunction sets the bit for the current address space in the address space map. If this is the first CICS region to start, it invokes IEAVG700 to initiate message broadcasting. DISCONNECT subfunction unsets the bit for the current address space in the address space map. If this is the last CICS region to finish, it invokes IEAVG700 to terminate message broadcasting.
DFHSSMGP	Message routine for DFHSSIN.
DFHSSMGT	Message table for DFHSSIN.
DFHSSWT	Router module for the console message handler. Calls DFHSSWTO for messages beginning with DFH. Calls DFHSSWTF for messages that echo MODIFY commands.
DFHSSWTF	Suppresses passwords from the echoed copies of MODIFY CICS commands that contain signon passwords.
DFHSSWTO	Inserts the applid into all DFH messages issued under a TCB with a valid AFCB.

Exits

There are no user exits in the subsystem interface support.

Trace

No tracing is performed by the subsystem interface support.

External interfaces

Module DFHSSIN invokes the MVS module IEEMB878 to read its initialization parameters from SYS1.PARMLIB.

Module DFHSSGC invokes the MVS module IEAVG700 to start and stop console message subsystem broadcasting.

Modules DFHCSVC and DFHSSEN use the IEFSSREQ interface to communicate with the CICS subsystem.

Chapter 47. Subtask control

Subtask control is the interface between a CICS task and a subtask. It avoids suspending CICS execution, and improves the response time.

This function is invoked by the DFHSK macro with the following calls:
- CTYPE=PERFORM activates an exit routine under a new TCB.
- CTYPE=WAIT waits for subtask to complete.
- CTYPE=RETURN returns control to the main CICS TCB.

Design overview

Some synchronous operating system requests issued by CICS modules could cause CICS to be suspended until the requests had completed. To avoid the resulting response-time degradation, certain requests are processed by the general-purpose subtask control program, DFHSKP. A CICS module calls DFHSKP to execute a routine within the module under a subtask of the operating system.

DFHSKP does the following:
- Schedules a subtask to execute a routine (called an SK exit routine)
- Allows an SK exit routine to wait on an event control block (ECB) of the operating system
- Manages subtask creation, execution, and termination
- Handles program checks or abends within the SK exit routine.

DFHSKP consists of the DFHSKM, DFHSKC, and DFHSKE programs.

DFHSKM (subtask manager program)

A DFHSK macro invokes DFHSKM to cause a routine to be executed under a subtask of the operating system. DFHSKM chooses a subtask to execute the request unless the caller has specified a particular subtask.

DFHSKM determines whether the subtask is inoperative, not started, or running. The subtask is called inoperative if it has terminated itself, or could not be attached. If the subtask is inoperative and the user coded SYNC=YES in the DFHSK macro, the request is processed synchronously; that is, DFHSKM executes the request under the CICS task control block (TCB).

If the subtask has not started, DFHSKM attaches a CICS task specifying the entry point of DFHSKC to execute. DFHSKM then waits on an ECB in the subtask control area (SKA) for the subtask and continues when the ECB is posted by DFHSKC, indicating that the subtask has been initialized.

DFHSKM then creates a work queue element (WQE) that represents the work to be performed under a subtask. The WQE is added to the work queue for the subtask. When the work ECB of the subtask is posted, signaling work to do, DFHSKM issues a wait on the work-complete ECB in the WQE. This ECB is posted when the WQE has been processed by the subtask. DFHSKM returns control to the caller, indicating the outcome of the processing.

If the subtask processing the WQE fails before completion, DFHSKM is informed and attempts to execute the request synchronously if the caller so specified.

When CICS terminates, it issues a DFHSK CTYPE=TERMINATE macro to terminate the subtasking mechanism. DFHSKM sets a flag in each subtask control area (in DFHSKP static storage) indicating that the subtask should terminate. DFHSKM then posts the subtask work ECB to signal the subtask to examine this flag.

DFHSKM is also invoked by deferred work element (DWE) processing.

When DFHSKM decides to process a WQE synchronously, control is passed to the routine specified by the caller. This routine may not complete normally and, so that DFHSKM does not lose the WQE because the task abended, it creates a DWE containing the address of the WQE. If the task abends, the DWE processor adds the WQE to the free queue.

DFHSKC (subtask control program)

DFHSKM invokes DFHSKC using the DFHKCP attach logic to start a subtask of the operating system, and wait for its completion. DFHSKM passes the address of the subtask control area in the facility control area address (TCAFCAAA) in the TCA.

DFHSKC issues an EXEC CICS GETMAIN for shared storage to pass to the subtask for use as its automatic storage. The length required is in a field in DFHSKE containing the automatic storage requirements. DFHSKC issues the ATTACH macro with the ECB option to attach the operating system subtask, and passes the address of the subtask control area.

DFHSKC issues the CICS SVC to authorize the TCB of the subtask to use the SVC.

DFHSKC issues a KC wait on the attach ECB. The module is suspended until subtask termination, when the ECB is posted. On termination, the subtask puts a return code in the subtask control area.

When the subtask completes, DFHSKC cleans up the subtask work queue. It then frees the automatic storage and terminates.

DFHSKC writes messages to CSMT from this module if it was unable to attach a subtask of the operating system subtask, or the subtask indicated that its termination was not normal.

DFHSKE (subtask exit program)

When the subtask manager DFHSKM, executing on behalf of a CICS task, decides that a subtask is to be started, it attaches a CICS task using the DFHKC ATTACH macro and specifying the entry point of DFHSKCNA. This CICS task attaches the subtask and waits for subtask completion by means of the ECB parameter coded in the ATTACH macro.

The ATTACH macro specifies an entry point in DFHSIP (known to MVS by an IDENTIFY macro issued in DFHSIP). DFHSIP then branches to the entry point of DFHSKE, whose address is in the subtask control area.

Note: DFHSIP remains in storage after initialization has completed.

The subtask reverses the order of the in-progress queue to service requests on a first-come, first-served basis. It then loops round the in-progress queue and, for each WQE, branches to the program specified in the WQE (the SK exit routine).

The exit routine returns control to DFHSKE, either indicating that the exit routine has completed by issuing a DFHSK CTYPE=RETURN macro or requesting that execution of the SK exit is suspended until an ECB specified by the exit is posted by some component of the operating system.

When a return is requested, the ECB in the WQE is posted, causing the dispatcher domain to resume the CICS task that was waiting for the SK exit to be complete. When a wait is requested, the WQE is added to the waiting queue, which is processed later.

When all WQEs in the in-progress queue have been processed, DFHSKE examines the waiting queue. If any WQEs are on this queue, their ECB addresses are inserted into an operating-system multiple-wait queue. The subtask work ECB (posted when a WQE is added to the work queue) is put at the top of this multiple-wait list. An operating-system multiple-wait is then issued.

When the subtask regains control, an ECB has been posted. This can be because more work has arrived or because an ECB belonging to an exit routine has been posted.

The WQEs on the waiting queue are scanned, and those whose ECB has been posted are moved to the in-progress queue, with a flag on indicating that an SK exit routine is to be resumed.

Control returns to the beginning of this program which examines the work queue and proceeds as described earlier.

DFHSKE handles program checks and operating system abends. If an abend exit is driven when processing a WQE, that WQE is blamed and processing of it terminates. The CICS task requesting the processing is informed of the problem.

If an abend exit is driven when a WQE is not being processed, it is assumed to be a problem in the subtasking program. The abend is handled, and a count of failures is increased. When a threshold is reached, the subtask terminates.

The MVS exits are ESTAE and SPIE.

For normal termination, DFHSKE loops, processing WQEs and waiting when there is no work to do. The subtask checks a flag in the subtask control area to see if it has been requested to terminate. If the flag is set, the subtask terminates, indicating normal termination by setting a response code in the subtask control area for the attacher, DFHSKC.

Abnormal termination may occur when the error threshold has been reached. The subtask terminates, but sets an error return code in the subtask control area for the attacher to see. The attacher, DFHSKC, then cleans up any outstanding WQEs on the subtask queues.

Control blocks

This function has the following control blocks:

- SK static storage contains pointers to free work queue elements (WQEs) and to work queue elements.
- SKRQLIST is the parameter area passed to DFHSKP on a request. It contains the address of the code to be executed, and the address of the ECB.
- DFHSKWPS is the WKE structure containing the address of the next WQE in the chain, the contents of the parameter field from CTYPE=PERFORM, the save area for registers, and the work-complete ECB.
- DFHSKAPS is the subtask control area. Each instance of this control block describes the state of one subtask and contains the address of automatic storage to be used by DFHSKE, pointers to the WQE used by the subtask, the current WQE being processed, and the ECB for work and completion.

See the *CICS Data Areas* manual for a detailed description of these control blocks.

Modules

Module	Function
DFHSKC	The subtask control program is invoked by DFHSKM to start up a subtask of the operating system
DFHSKE	The general-purpose multitask program is executed as a subtask of the operating system
DFHSKM	The subtask manager program causes the routine to execute under a subtask.

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for this function:
- AP 00DE, for which the trace level is AP 1.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

External interfaces

The following external calls are made by DFHSKC:

MVS ATTACH	To attach a new TCB
MVS DETACH	To detach a TCB
MVS POST	To post a CICS TCB.

The following external calls are made by DFHSKE:

MVS ESTAE	To establish an error exit
MVS WAIT	To synchronize with the TCB
MVS SETRP	To retry after a failure.

Chapter 48. Syncpoint program

This allows the user to specify logical units of work by means of **syncpoints**. Any processing performed between syncpoints (provided the resources are declared as recoverable) can be reversed in the event of an error; but ***after*** a given syncpoint has been reached, the processing performed ***before*** that syncpoint cannot be reversed.

A syncpoint is also taken automatically at the end of each task.

Design overview

The syncpoint program works in conjunction with the Recovery Manager domain to provide the user with the ability to establish points in application programs at which all recoverable updates are committed. (The user can, at any time, back out any uncommitted changes by means of the rollback function.)

The syncpoint interface is provided by the DFHSPP module. DFHSPP is invoked, via the EXEC Interface module DFHEISP, when an application program issues an EXEC CICS SYNCPOINT or SYNCPOINT ROLLBACK command. It is also called from other CICS modules, such as DFHMIRS.

Further important information about syncpoint processing is given in Chapter 26, “Function shipping,” on page 277 and Chapter 99, “Recovery Manager Domain (RM),” on page 1061.

DFHSPP implements syncpoint calls by in turn calling the Recovery Manager domain with DFHRMUWM COMMIT_UOW or BACKOUT_UOW requests. RM calls its clients with prepare, commit, start backout etc. calls. One of RM's clients is ’APUS’, serviced by module DFHAPRC. Depending on the call from RM DFHAPRC calls DFHSPP or DFHDBP to process Deferred Work Elements (DWEs). DWEs provide a mechanism whereby resource owners can record their need to perform actions at a syncpoint. Most resource owners provide their own RM client routines, but a few, such as interval control, use DWEs.

Note that the implicit syncpoint or backout performed at task termination is effected by a direct call to the RM domain, not by issuing a DFHSP macro.

Task-related user exit resynchronization

The purpose of task-related user exit resynchronization is to allow a resource manager to ask CICS for the resolution of UOWs about which it is in-doubt. Task related user exit resynchronization is called as a result of an EXEC CICS RESYNC command to restore the CICS end of the thread that was interrupted by the failure of the connection with the resource manager.

DFHRMSY is passed a parameter list by DFHERMRS which consists of the following: rmi entryname (8 bytes) - the name of the TRUE to be called for resync. rmi qualifier (8 bytes) - the qualifier to the name of the TRUE to be called for resync. uowid (8 bytes) - the id of the UOW to be resynchronized resync type (1 byte) - a flag indicating whether this is a resync as a result of an EXEC CICS RESYNC command or due to a Recovery manager domain unshunt.

DFHRMSY's job is to call the named TRUE with a resync call giving the resolution of the named UOW. The resolution can be commit, backout, should not be indoubt or lost to initial start. (Lost to initial start means that a START=INITIAL has been performed subsequent to the indoubt UOW being created. Initial start clears the log and the catalog meaning that Recovery Manager has no knowledge of the UOW.)

In order to find the outcome of the UOW, DFHRMSY issues a INITIATE_RECOVERY call to Recovery manager domain for the named UOW, which returns the UOW status. DFHRMSY then builds the resync plist to pass to the TRUE, and calls the TRUE using a DFHRMCAL macro. On return from the TRUE, if the TRUE returns an OK response indicating that it has successfully resynced with its resource manager, then DFHRMSY issues a TERMINATE_RECOVERY call to RECOVERY manager domain specifying

FORGET(YES). This tells RM domain it can remove this TRUE's involvement in the UOW. If no other components or TRUEs are waiting resync for the UOW, then RM domain will delete it's knowledge of the UOW. If the TRUE does not return with an ok response, FORGET(NO) is specified on the TERMINATE_RECOVERY call, and RM domain retains this UOW for this TRUE. A subsequent resync will be required.

Control blocks

This section describes the control blocks used by the syncpoint program:
- Deferred work element (DWE)

See the *CICS Data Areas* manual for a detailed description.

Deferred work element (DWE)

A deferred work element (DWE) is created and placed on a DWE chain to save information about actions that must be taken when the unit of work terminates. These actions may depend upon whether the UOW commits or backs out.

DWEs are created by CICS control modules, and chained off field TCADWLBA in the task’s TCA using DWECHAN as the chain field. The module that creates a DWE inserts the entry address of a DWE processor in field DWESVMNA of that DWE. Control is passed to this DWE processor by the syncpoint program at the end of the task or UOW.

DWEs can be used for work to be done before or after the syncpoint is logged or in the event of transaction backout.

The layout of DWEs is defined by the DFHDWEPS structure and by the DFHDWEDS assembler DSECT.

Modules

DFHSPP, DFHAPRC, DFHDBP

DFHSPP

DFHSPP can be invoked by the following macros:

DFHSP TYPE=USER	Take a syncpoint
DFHSP TYPE=ROLLBACK	Roll back the current unit of work
DFHSP TYPE=PHASE_1	Do DWE processing for prepare
DFHSP TYPE=PHASE_2	Do DWE processing for commit

When DFHSPP is called by means of a DFHSP TYPE=USER or TYPE=ROLLBACK macro the request is converted into a call to the Recovery Manager domain to commit or backout the current UOW. If the RM request fails SPP calls DFHAPAC to select an abend code corresponding to the failure reported by RM (for example ASP1 for an in-doubt failure) and, in most cases, issues a PC ABEND with this abend code.

In the case of a commit or backout failure, however, no PC ABEND is issued and the transaction continues normally. In these cases CICS has, for the present, been unable to bring all local resources to the committed state for this unit of work. It has recorded any data necessary to re-attempt this at some later time, and has retained any locks necessary to preserve data integrity until then.

When DFHSPP is called by means of a DFHSP TYPE=PHASE_1 or TYPE=PHASE_2 macro SPP processes any DWEs in the DWE chain (TCADWLBA). The TYPE=PHASE_1 call is issued by DFHAPRC in response to an RM prepare or end_backout request. For each DWE in the chain that is not marked as cancelled (DWECNLM ON) or phase_2 only (DWEPHS2 ON) the DWE processor (entry address DWESVMNA) is called. In the prepare case SPP collects ’votes’ and may return a YES, NO or READ-ONLY vote to its caller. Also, if necessary, a DL/I TERM call is issued to allow DFHDLI to perform end-of-UOW actions. The TYPE=PHASE_2 call is issued by DFHAPRC in response to an RM commit or shunt request. For each DWE in the chain that is marked phase 2 and not cancelled the DWE processor is called. In the shunt case any DWE that is marked for shunting (DWESHUNT ON) is retained in the DWE chain. All other DWEs are freed.

DFHDBP

DFHDBP is link-edited with DFHAPRC and is called by DFHAPRC in response to an RM start_backout request. For each DWE in the task's DWE chain that is not marked cancelled it marks the DWE as 'backout' (DWEDYNB ON). For any BMS DWE it issues a DFHBMS TYPE=PURGE request to discard the incomplete message, otherwise it calls the DWE processor then marks the DWE as cancelled.

DFHAPRC

DFHAPRC is the module which provides the gate for the 'APUS' Recovery Manager client. It provides keypoint and restart support for user written log records, which is described elsewhere, and syncpoint support where it serves as a receiver for RMRO calls from the RM domain for prepare, commit, etc. which it converts into appropriate calls to SPP or DBP described above.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:
- AP 00CB, for which the trace level is AP 1.
- AP D8xx, for which the trace level is AP 1.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 49. System dump formatting program

The system dump formatting program is for use on MVS system dump (SYS1.DUMP) data sets that record system dumps requested by CICS via the MVS SDUMP macro.

The program is invoked via the interactive problem control system (IPCS). You can use IPCS either interactively or from an MVS batch job.

The CICS-supplied sample system dump formatting program for use with CICS Transaction Server for z/OS, Version 3 Release 1 control blocks is called DFHPD640.

For further information about the system dump formatting programs, about using IPCS to format and analyze CICS dumps, and about the dump exit parameters available, see the *CICS Operations and Utilities Guide*.

Design overview

The system dump formatting program produces a formatted listing of CICS control blocks grouped within functional area. CICS dump exit parameters can be specified on the IPCS VERBEXIT subcommand to indicate whether the control block output is to be produced or suppressed for each functional (component) area. Summary reports are available for certain of the functional areas, and the dump exit parameters can also indicate whether these are to be produced or suppressed.

Modules

Module	Function
DFHAIDUF	Autoinstall terminal model manager formatter
DFHAPTRA	Application domain multiregion operation trace interpreter
DFHAPTRB	Application domain extended recovery facility trace interpreter
DFHAPTRC	Application domain user exit trace interpreter
DFHAPTRD	Application domain trace interpreter
DFHAPTRE	Application domain data tables trace interpreter
DFHAPTRF	Application domain SAA Communications and Resource Recovery interfaces trace interpreter
DFHAPTRG	Application domain ZC exception and VTAM exit trace interpreter
DFHAPTRI	Application domain trace interpretation router
DFHAPTRJ	Application domain ZC VTAM interface trace interpreter
DFHAPTRL	Application domain CICS OS/2 LU2 mirror trace interpreter
DFHAPTRN	Application domain autoinstall terminal model manager trace interpreter
DFHAPTRO	Application domain LU6.2 application request logic trace interpreter
DFHAPTRP	Application domain program control trace interpreter
DFHAPTRR	Application domain partner resource manager trace interpreter
DFHAPTRS	Application domain DFHEISR trace interpreter
DFHAPTRV	Application domain DFHSRP trace interpreter
DFHAPTRW	Front End Programming Interface feature trace interpreter
DFHAPTR0	Application domain old-style trace entry interpreter
DFHAPTR2	Application domain statistics trace interpreter
DFHAPTR4	Application domain transaction manager trace interpreter
DFHAPTR5	Application domain file control trace interpreter
DFHAPTR6	Application domain DBCTL DL/I trace interpreter
DFHAPTR7	Application domain LU6.2 transaction routing trace interpreter
DFHAPTR8	Application domain security trace interpreter
DFHAPTR9	Application domain interval control trace interpreter
DFHCCDUF	CICS catalog formatter

System dump formatting program

Module	Function
DFHCCTRI	CICS catalog trace interpreter
DFHCPDUF	SAA Communications and Resource Recovery interfaces formatter
DFHCSDUF	CSA and CSA optional features list formatter
DFHDBDUF	DBCTL and remote DL/I dump formatter
DFHDDDUF	Directory manager formatter
DFHDDTRI	Directory manager trace interpreter
DFHDMDUF	Domain manager formatter
DFHDMTRI	Domain manager trace interpreter
DFHDSDUF	Dispatcher domain formatter
DFHDSTRI	Dispatcher domain trace interpreter
DFHDUDUF	Dump domain formatter
DFHDUF	Formatting router
DFHDUFUT	Service functions routine
DFHDUTRI	Dump domain trace interpreter
DFHERDUF	Error message index processor
DFHFCDUF	File control formatter
DFHFRDUF	File control recoverable work elements formatter
DFHICDUF	Interval control formatter
DFHIPCSP	Table of CICS entries for the IPCS exit control table
DFHIPDUF	Kernel stack internal procedure formatter
DFHKEDUF	Kernel domain formatter
DFHKELOC	Routine for locating domain anchors
DFHKETRI	Kernel domain trace interpreter
DFHLDDUF	Loader domain formatter
DFHLDTRI	Loader domain trace interpreter
DFHLMDUF	Lock manager formatter
DFHLMTRI	Lock manager trace interpreter
DFHMEDUF	Message domain formatter
DFHMETRI	Message domain trace interpreter
DFHMNDUF	Monitoring domain formatter
DFHMNTRI	Monitoring domain trace interpreter
DFHMRDUF	Multiregion operation formatter
DFHNXDUF	Control block index processor
DFHPADUF	Parameter manager formatter
DFHPATRI	Parameter manager trace interpreter
DFHPDKW	Input parameter string validation routine
DFHPDX1	Control program
DFHPGDUF	Program manager formatter
DFHPGTRI	Program manager trace interpreter
DFHPRDUF	Partner resource manager formatter
DFHPTDUF	Program control table formatter
DFHRMDUF	Resource recovery manager formatter
DFHSMDUF	Storage manager formatter
DFHSMTRI	Storage manager trace interpreter
DFHSNTRI	Application domain signon trace interpreter
DFHSSDUF	Static storage area formatter
DFHSTDUF	Statistics domain formatter
DFHSTTRI	Statistics domain trace interpreter
DFHSUDUF	Dump domain summary formatter
DFHSUTRI	Subroutine trace interpreter
DFHSZDUF	Front End Programming Interface feature dump formatter
DFHTCDUF	Terminal control formatter
DFHTDDUF	Transient data formatter
DFHTDTRI	Transient data trace interpreter

Module	Function
DFHTIDUF	Timer domain formatter
DFHTITRI	Timer domain trace interpreter
DFHTMDUF	Table manager formatter
DFHTRDUF	Trace domain formatter
DFHTRFFD	Trace entry data field formatter
DFHTRFFE	Trace entry formatter
DFHTRFPB	Routine to process blocks of trace entries
DFHTRFPP	Routine for selecting trace entries to be printed
DFHTRIB	Trace entry interpretation string builder
DFHTRTRI	Trace domain trace interpreter
DFHTSDUF	Temporary-storage formatter
DFHUEDUF	User exit formatter
DFHUSDUF	User domain dump formatter
DFHUSTRI	User domain trace interpreter
DFHXMDUF	Transaction manager domain formatter
DFHXMTRI	Transaction manager domain trace interpreter
DFHXSDUF	Security domain dump formatter
DFHXSTRI	Security domain trace interpreter
DFHXRDUF	Extended recovery facility (XRF) formatter
DFHZXDUF	XRF ZCP queue formatter

Exits

Global user exit points are not applicable to offline utilities.

Trace

Trace points are not applicable to offline utilities. However, the output obtained and any messages issued by the system dump formatting program may provide clues to problems associated with corrupted data.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

External interfaces

The following external calls are used by the system dump formatting program:

- MVS GETMAIN and FREEMAIN for storage management
- OPEN SVC to open DFHSNAP
- CLOSE SVC to close DFHSNAP
- MVS IPCS service routines.

Chapter 50. System recovery program

The system recovery programs, DFHSR1, DFHSRP, and DFHSRLI, together form the default CICS recovery routine for the application (AP) domain. This routine is, in particular, the recovery routine for program checks, operating system abends, and runaway tasks that occur in user application code.

Design overview

The CICS kernel intercepts program checks, runaway tasks, operating system abends and some other internal errors for all CICS domains. The kernel then selects which CICS recovery routine to pass control to. The selected recovery routine can then process the error as appropriate.

The DFHSR1 module is the default recovery routine for the application domain. It receives control if any of the above errors occur in CICS system application programs, user application programs and some CICS nucleus modules. It processes internal errors itself but, when dealing with program checks, operating system abends, and runaway task abends, it calls the DFHSRP module. The DFHSRP module, in turn, converts the error into a transaction abend, if possible; if not possible, it terminates CICS. The DFHSRP module uses subroutines in DFHSRLI.

The transaction abend codes that may be issued are:

AEYD
error detected by command protection

AICA task runaway

AKEF domain gate not active

AKEG kernel stack storage GETMAIN failure.

ASRA program check

ASRB
operating system abend

ASRD illegal macro call or attempt to access the CSA or TCA

ASRK
TCA not available

xxxx as set by issuers of deferred abend

The processing associated with each of these abends is described in “Error handling” on page 376.

For further information about the abends, see the *CICS Messages and Codes* manual.

System recovery table

Associated with DFHSRP is the system recovery table (SRT). This is a table that the user can provide, containing operating system abend codes. It controls whether CICS recovers from program checks and operating system abends in noncritical code.

You specify the name of the system recovery table by the SRT system initialization parameter, as either SRT=NO or SRT=xx, where xx is the two-character suffix of the SRT:

- If NO is coded, CICS does not recover from program checks or operating system abends, and terminates if one occurs.
- If a suffix is coded, CICS attempts to recover from all types of program check, but can only recover from an operating system abend if the abend code appears in the SRT identified by the suffix (for example, DFHSRT1A where 1A is the suffix). If the abend code is not in the SRT, CICS terminates.

For information about how to create the SRT, see the *CICS Resource Definition Guide*.

Recovery initialization

The DFHSII1 module calls the DFHSR1 module during AP Domain initialization. The DFHSR1 module tells the Kernel that it is the default recovery routine for the AP domain and adds the ABAB gate.

If any error occurs when informing the kernel, CICS is terminated with message DFHSR0605 and a system dump because it is not possible to run CICS without AP domain recovery.

Error handling

The DFHSR1 module gets control from the kernel or from other AP domain modules. It decides whether it is dealing with an internal error or an external error such as a program check. Internal errors are dealt with by exiting from the recovery environment and issuing the appropriate kernel call. If either of the DFHXFP or DFHEMS modules has caused a program check, the DFHSR1 module exits from the recovery environment and passes control to DFHXFP or DFHEMS. All other external errors are passed on to the DFHSRP module. If control returns from the DFHSRP module, DFHSR1 issues a transaction abend. If control returns from the abend call, it is because the XPCTA exit has requested retry; in which case, DFHSR1 restores the registers etc and branches to the resume address.

The DFHSRP module makes an exception trace entry, ensures it is running on the QR TCB and then deals with one of the following:

- Program check (see "Program check")
- Operating system abend (see "Operating system abend" on page 377)
- Runaway task (see "Runaway task" on page 378)
- Kernel gate error (see "Kernel gate error" on page 378)
- Deferred abend. (see "Deferred abend" on page 378).

Note: The kernel recovery environment is terminated very soon after DFHSRP receives control. This ensures that DFHSRP gets driven again if a subsequent error occurs in DFHSRP itself (rather than the kernel percolating the error to the next kernel stack entry). DFHSRP is therefore in a position to detect such recursive errors, and can take the appropriate action.

If DFHSRP can abend the transaction, it builds a Transaction Abend Control Block (TACB) to describe the abend. The TACB is a task-lifetime control block that records details of a transaction abend. This TACB may be used by the rest of AP domain that needs information about the abend. DFHSRP builds the TACB, rather than letting Program Control build it as part of DFHPC TYPE=ABEND processing. This enables DFHSRP to include extra information in the TACB that would otherwise be lost, such as GP registers, PSW, and FP registers at the time of the error.

Program check

The following processing takes place for a program check, in the order given:

1. If this program check occurred while DFHSRP was in the middle of processing a previous program check, then CICS is terminated with message DFHSR0602 and a system dump. Otherwise DFHSRP may get caught in a recursive loop.
2. If this program check occurred while DFHSRP was in the middle of processing an operating system abend, then CICS is terminated with message DFHSR0615 and a system dump. This traps program checks in global user exit XSRAB.
3. If DFHEIP hired gun checking caused the program check, create an abend record for abend code AEYD and return to DFHSR1.
4. If the program check was an 0C4 protection exception, DFHSRP diagnoses the 0C4 further in order to establish whether it was caused by an attempt to access or overwrite CICS-managed protected storage. Such storage is as follows:
   - The fetch-protected dummy CSA block

- The CDSA
- The ECDSA
- The ERDSA.
- The EUDSA.
- The RDSA.
- The UDSA.

Of the above, it should be noted that one can only 0C4 on the CDSA or ECDSA if storage protection is active, while 0C4 on the UDSA or EUDSA can only be obtained if transaction isolation is active.

This diagnosis is accomplished by disassembling the failing instruction, and examining the instruction operands in conjunction with the execution conditions at the time of the 0C4 (such as execution key). If the dummy CSA caused the 0C4 (that is, an attempt was made to access the CSA or TCA, or an illegal macro call was issued), message DFHSR0618 is issued. If a DSA caused the 0C4, message DFHSR0622 is issued.

5. If the SRT=NO system initialization parameter was specified, you have disabled recovery, and CICS terminates with message DFHSR0603 and a system dump.
6. If a CICS system task was in control at the time of the program check, indicated by a non-numeric transaction number, CICS is terminated with message DFHSR0601 and a system dump.
7. Some special processing is performed which applies only to PL/I programs.
8. DFHSRLI is called to determine the following information:
   - The program in which the program check occurred
   - The offset in that program
   - The execution key.
9. The results of the diagnosis (program, offset, execution key, and, if an 0C4 abend, any “hit” DSA) are output in an exception trace.
10. Message DFHAP0001 or DFHSR0001 is issued and a system dump is taken. (See also “System dump suppression” on page 379.)

    Whether message DFHAP0001 or DFHSR0001 is issued is governed by the execution key at the time of the program check. If the program was running in user key, message DFHSR0001 is issued; otherwise, message DFHAP0001 is issued.
11. Finally, DFHSRP creates an abend record and returns to DFHSR1.

Operating system abend

The following processing takes place for an operating system abend, in the order given:

1. If this abend occurred while DFHSRP was in the middle of processing a previous operating system abend, then CICS is terminated with message DFHSR0612 and a system dump. Otherwise, DFHSRP may get caught in a recursive loop.
2. If the SRT=NO system initialization parameter was specified, you have disabled recovery, and CICS terminates with message DFHSR0606. A system dump may be taken, if specified on the operating system abend.
3. If the SRT=xx system initialization parameter was specified, DFHSRP searches the SRT with the suffix xx (that is, DFHSRTxx) for the abend code. If it does not find the abend code, CICS terminates with message DFHSR0606. A system dump may be taken, if specified on the operating system abend.
4. When the abend code has been located, the next check is to see if the operating system abend occurred in a CICS system task, indicated by a non-numeric transaction number. If so, CICS terminates with message DFHSR0613 and a system dump.
5. Otherwise, the default decision is to abend the transaction with code ASRB. However, you can modify this decision by coding a global user exit program at exit point XSRAB. In addition to performing any processing that might be required for particular operating system abends, the XSRAB exit point allows you to specify whether to:
   - Terminate CICS
   - Abend the transaction ASRB

- Abend the transaction ASRB, but cancel any active HANDLE ABEND exits.

6. If you choose to terminate CICS, CICS terminates with message DFHSR0606. A system dump may be taken, if specified on the operating system abend.
7. DFHSRLI is called to determine the following information:
   - The program in which the program check occurred
   - The offset in that program
   - The execution key.
8. The results of the diagnosis (program, offset, and execution key) are output in an exception trace.
9. Message DFHAP0001 or DFHSR0001 is issued and a system dump is taken. (See also "System dump suppression" on page 379.)

   Whether message DFHAP0001 or DFHSR0001 is issued is governed by the execution key at the time of the program check. If the program was running in user key, message DFHSR0001 is issued; otherwise, message DFHAP0001 is issued.
10. Finally, DFHSRP The DFHSRP module creates an abend record with abend code ASRB returns to DFHSR1.

Runaway task

One of the following processing options takes place for a runaway task:

- If this runaway task occurred while DFHSRP was in the middle of processing an operating system abend, CICS terminates with message DFHSR0612 and a system dump. This traps runaway tasks caused by errors in global user exit XSRAB.
- Otherwise, the DFHSRP module creates an abend record with abend code AICA and returns to DFHSR1.

Kernel gate error

One of the following processing options takes place for a kernel gate error:

- If this error occurred while DFHSRP was in the middle of processing an operating system abend, CICS terminates with message DFHSR0612 and a system dump. This traps kernel gate errors from XPI calls in global user exit XSRAB.
- Otherwise, the DFHSRP module issues message DFHAP0001, creates an abend record with abend code AKEF, and returns to DFHSR1.

kernel stack GETMAIN error

The processing that takes place for a kernel stack GETMAIN error is identical to the processing for a kernel gate error, except that the transaction is abended with abend code AKEG.

Deferred abend

The DFHSRP module creates an abend record using the abend code set by the code that issued the deferred abend and returns to DFHSR1.

DFHSRLIM interface

This interface is used to call program DFHSRLI. It provides the following functions for DFHSRP:

INVOKE_XSRAB

This function invokes global user exit XSRAB if active, passing to it structure SRP_ERROR_DATA which contains details of the operating system abend that occurred. The abend recovery option selected by the exit is returned, which is either to terminate CICS, abend the transaction ASRB, or abend the transaction ASRB and cancel any active abend exits.

DIAGNOSE_ABEND

This function diagnoses a program check, operating system abend, or other error, to establish the location of the error. It returns the program in which the error occurred, the offset within that program, and whether the error occurred in CICS or user application code. (A decision based on the execution key; user key implies user application code.)

System dump suppression

When message DFHAP0001 or DFHSR0001 is issued before the transaction is abended with ASRA, ASRB, ASRD, AKEF, or AKEG, the default is to take a system dump with dumpcode AP0001 or SR0001 respectively. Message DFHSR0001 is issued if CICS is running with storage protection active and is running in user key at the time of the error; otherwise, message DFHAP0001 is issued.

Therefore, it is possible to suppress the system dumps taken for errors occurring in code that is being run in user key (user application code), while retaining system dumps for errors occurring in code that is being run in CICS key (CICS code), by adding SR0001 to the dump table specifying that no system dump is to be taken.

Note that the XDUREQ Global User Exit can be used to distinguish between AP0001 situations in application and non-application code. This allows selective dump suppression when storage protection is not active or when it is active but some applications run in CICS key.

Modules

Module	Function
DFHSRP	Called by DFHSR1 to process program checks, operating system abends, runaway tasks, and so on.
DFHSRLI	Provides functions for DFHSRP, via the DFHSRLIM interface.
DFHSR1	The default recovery routine for the AP Domain.

Exits

There is one global user exit point in DFHSR1: XSRAB. This exit can be called if an operating system abend has occurred and the abend code is in the SRT.

For further information about using the XSRAB exit, see the *CICS Customization Guide*.

Trace

The following trace point IDs are provided for DFHSRP and DFHSRLI:
- AP 0701, for which the trace entry level is AP 2
- AP 0702, for which the trace entry level is AP 2
- AP 0780, for which the trace entry level is Exc
- AP 0781, for which the trace entry level is Exc
- AP 0782, for which the trace entry level is Exc
- AP 0783, for which the trace entry level is Exc.
- AP 0790, for which the trace entry level is Exc
- AP 0791, for which the trace entry level is Exc
- AP 0792, for which the trace entry level is Exc
- AP 0793, for which the trace entry level is Exc.
- AP 0794, for which the trace entry level is Exc
- AP 0795, for which the trace entry level is Exc
- AP 0796, for which the trace entry level is Exc
- AP 0797, for which the trace entry level is Exc.
- AP 0798, for which the trace entry level is Exc

- AP 0799, for which the trace entry level is Exc.
- AP 079A, for which the trace entry level is Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 51. System spooler interface

A system programmer can communicate with the local system spooler and, consequently, with other system spoolers via the system spooler network facilities. The system spooler interface single-threads its input, and it is the user's responsibility to see that all transactions get the chance to run. One high-priority transaction should not use the interface exclusively.

Further information about the system spooler interface is given in the *CICS Application Programming Reference*.

Design overview

The system spooler interface program opens a system spooler file for either input or output, reads or writes a file, and closes a file. These functions are for system programmer use. The input is single-threaded, so only one transaction can use it at a time.

An application can send files to a remote location by specifying the node of the location, and the userid (or external writer name) of the user at that location. To retrieve a file at the remote location, you specify the external writer name, and you can then retrieve reports from that writer. For security reasons, the external writer name must begin with the same four characters as the CICS applid. The remote system to which a file or report is sent, or from which it is received, must have JES under MVS, or VM.

System spooler interface modules

The SPOOLOPEN command dynamically allocates input or output files using the CICS SVC, and an application control block (ACB) is opened to process the file. For an input file, the IEFSSREQ macro is also issued to determine which file to process. The SPOOLREAD or SPOOLWRITE commands cause GETs or PUTs to be issued using the ACB. The SPOOLCLOSE command dynamically deallocates a file, and causes it to be either transmitted or deleted. All processing which could cause CICS to be suspended is performed under an operating system subtask which is initiated by subtask control, DFHSKP.

DFHPSPST runs under CICS, but DFHPSPSS, and modules called as a result, run under the subtask.

Normal flow

When a system spooler interface command is executed, the normal sequence of invocation of modules is:
1. DFHEIP
2. DFHEPS
3. DFHPSP
4. DFHPSPSS
5. DFHPSPST
6. DFHPSSVC.

DFHPSP is called by:
- Application programs via DFHEPS issuing the DFHPS macro.
- Syncpoint program and dynamic transaction backout program to the deferred work element (DWE) module (DFHPSPDW). The entry address of DFHPSPDW is stored in the DWE. DFHPSPDW then calls DFHPSPST via DFHPS.

Abnormal flow

If a user transaction terminates without issuing a SPOOLCLOSE command, DFHPSPDW is invoked to process a DWE that was set up when the SPOOLOPEN command was processed. This closes the file in the usual way.

Modules

Module	Name
DFHEIP	DFHEIP initializes the EXEC interface structure (EIS) and then invokes the application program. Each EXEC CICS command invokes DFHEIP (nucleus) which in turn invokes the appropriate interface processor. DFHEIP also returns information to the application program through EIB (within EIS).
DFHEPS	DFHEPS is the link between DFHEIP and the JES interface program, DFHPSP.
DFHPSP	DFHPSP is the system spooler interface control module.
DFHPSPCK	DFHPSPCK is the JES interface termination processor.
DFHPSPDW	DFHPSPDW is the DWE processor.
DFHPSPSS	The system spooler interface subtask module attaches a subtask to check that a writer name and a token have been supplied. It opens and closes JES data sets, reads a record, and writes a record.
DFHPSPST	DFHPSPST is the JES interface controller.
DFHPSSVC	DFHPSSVC is the system spooler interface module that retrieves a data set name for a given external writer name, dynamically allocates it, and returns its DDNAME.

Exits

No global user exit points are provided for this interface.

Trace

The following point ID is provided for this interface:
- AP 00E3, for which the trace level is AP 1.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 52. Table manager

The table manager controls the locating, adding, deleting, locking, and unlocking of entries in certain CICS tables. These operations can be performed while CICS is running.

Design overview

Locating, adding, deleting, locking, and unlocking entries in tables such as the file control table (FCT), application file control table (AFCT), data set name block table (DSNT), and terminal control table (TCT) are performed by the table manager program, DFHTMP. Entries in these tables are also called “resources”. Because the structures of tables vary as entries are added or deleted, and a quick random access is required, a hash table mechanism is used to reference the table entries. In addition because fast access is needed for generic locates and ordered lists of entries, a getnext chain with a range table is used.

Hash table

The hash table is a set of pointers that are the addresses of directory elements of table entries. A directory element is a set of pointers; one of these pointers is the address of the table entry, the remaining pointers are the addresses of the next elements of various chains used in the different operations of the table manager. An example of a hash table is shown in Figure 75 on page 384.

The table manager logically combines the characters of the name of the resource, and transforms the result to give an integer that is evenly distributed over the hash table size.

When an entry is located or added, the table manager places it at the head of its chain. Thus frequently used entries tend to have the minimum search times.

If the hash chains become very long, the table manager creates a larger hash table if storage is available. The hash table is enqueued before and dequeued after the reorganization, so that no references to the table can be made during reorganization.

Note: Certain TMP hash tables are not reorganized because they are also used in VTAM SRB exits.

Range table and getnext chain

Some requests to TMP are not full key locates, but rather generic locates with a partial key. For example, requests to find all terminals whose Termid starts with two specified characters. To enable these requests, a getnext chain is maintained which orders all the directory elements alphabetically by key. There is also a ‘range table’ which holds pointers to certain elements along the getnext chain and a count of how many intermediate elements there are in each range.

This range table is hunted with a binary search to find the range in which a given key (full or partial) will reside, and then the getnext chain is used to find a match (if one exists) for the search condition.

A range will be split into two equal ranges if the number of intermediate elements rises above a threshold which depends on the number of ranges and the number of elements in the table. So the ranges are dynamic, and do not depend on any particular key distribution.

The number of ranges in the table is determined when the hash table is created, and if all the ranges are full, but a range should be split, a reorganization of the ranges takes place, which increases the range threshold by a factor of 2.

Secondary indexes

A separate hash table, called the secondary index, is created for certain TMP tables, which allows the same entry to be located by another key. In certain secondary indexes, the names do not need to be unique (whereas in the primary index the name is always unique). The secondary index entry is deleted at the same time the entry in the primary index is deleted.

For example, a secondary index is created for DSNAME blocks. This allows table entries to be accessed via secondary keys, using the DSNAME block number in the case of DSNAME blocks.

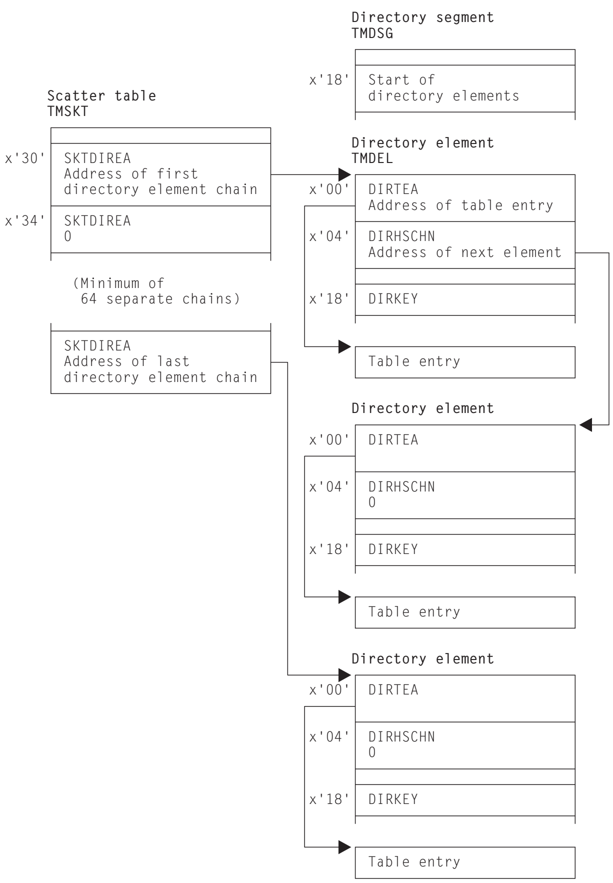


Figure 75. Example of a hash table

Certain tables also have aliases as distinct from secondary indexes. These are alternative names for the table entry, which can be used to locate a table entry. They exist in the same index as the primary name, and are not included in a getnext chain, rather they form an alias chain from the primary entry.

Functions of the table manager

The table manager performs the following functions:

Locate table entry
For a given name, find the address of the table entry.

Get next table entry
For a given name, find the address of the next table entry in collating sequence. This can be used repeatedly to find all entries in a range (or all elements in the whole table).

Add table entry
For a given table entry, add it into the table.

Quiesce a table entry
For a given name, mark its directory segment as busy.

Unquiesce a table entry
For a given name, remove its directory segment from the 'quiesce' state.

Delete a table entry
For a given name, delete it and any associated alias. The entry must have been quiesced first.

Create an index for a table Create a hash table of a given type.

Add a name into a secondary index
Given a primary name and a secondary name, add the names to the secondary index.

Add an alias name For a given name, assign an alias name.

Get next alias name For a given a name, find the next alias name (if any).

Lock a table entry For a given a name, assign a read lock to it.

Unlock a directory entry For a given a name, remove the associated read lock.

Reset lock slots For a given name, reset the lock slots.

Transfer lock to target task For a given a name and the address of a target TCA, transfer the read lock to the target task.

Process deferred work element
Make the changes made by the logical unit of work (LUW) visible at task syncpoint time.

Read locks

Read locks are used to prevent a table entry being deleted by the table manager.

A read lock is a fullword of storage. When DFHKCP attaches a task, it allocates storage for a number of local read locks; this storage is addressed by TCATMRLP in the TCA. Local read locks are not acquired for table entries that cannot be deleted.

Global read locks are used by the CICS modules that are executed independently of any task. They reside in the table manager static storage area (TMS) that is addressed by SSATMP in the static storage address list (SSA).

These locks are released by:
- an Unlock call,
- a Getnext call,
- a Reset call,
- the termination of the task,
- or a DWE call.

Read locks are always obtained against the primary index entry even if the request is against a secondary index or an alias.

Browse token

For Getnext requests on secondary indexes, a browse token is used to hold the name of the previously found entry. The token consists of the name found in the secondary index (which may not be unique) and the name in the primary index (which is unique).

The address of the directory entry cannot be used instead of this logical name because the entry may be returned unlocked, and so may be deleted when the next getnext request is received.

The getnext consists of locating the entry in the secondary index which has a the correct primary index, if it exists, and then moving forward in the getnext chain. If it does not, an entry with a matching secondary index name, but a higher primary index name is located, if one exists. If that also does not exist, an entry with a higher name in the secondary index is located. This requires that entries on the getnext chain for ordered both by secondary index name and also when identical secondary index names exist, by primary index name.

Quiesce state

A table entry is moved into quiesce state by a quiesce request if no read locks (including ones obtained by the issuing task) exist for the entry. When a table entry moves into quiesced state, it is unable to be located. Locating tasks can choose to ignore or wait for quiesced entries to be unquiesced or deleted.

If the quiesce request is performed with the commit option, the only ways to release the quiesced state are:
- Unquiesce
- Delete

For commit requests, the delete takes place immediately the request completes. Otherwise, if an entry is not deleted or unquiesced by the end of the UOW the TM DWE will unquiesce the entry. In this case, a delete does not take effect until the end of the UOW.

Finding FCT, or TCT entries in a partition dump

Figure 76 on page 387 shows the relationship of the table manager control blocks. A general procedure for finding the required table entries in a partition dump is as follows:

1. Find the CSA.
2. Find the CSA optional features list, CSAOPFL, from its address in field CSAOPFLA (offset X'C8') in the CSA.
3. Find the static storage area address list (SSA) from its address in field CSASSA (offset X'1C0') in the CSAOPFL.
4. Find the table manager static storage area (TMS) from its address in field SSATMP (offset X'14') in the SSA.
5. Look at TMS in the *CICS Data Areas* manual. The fields TMASKT1 through TMASKT24 hold the addresses of the hash tables for various control blocks. Find the hash table for the control block you are interested in:

```
TMASKT1 = reserved
TMASKT2 = reserved
TMASKT3 = reserved
TMASKT4 = addr of profile table (PFT) entries
TMASKT5 = addr of FCT entries
TMASKT7 = addr of local terminal (TCTE) entries
TMASKT8 = addr of remote terminal and connection (TCNT) entries
TMASKT9 = addr of local connection (TCTS) entries
TMASKT10 = addr of AFCT entries
TMASKT11 = addr of DSNAME entries (by name)
TMASKT12 = addr of DSNAME entries (by block ID)*
TMASKT13 = addr of partner resource table (PRT)
           entries
TMASKT14 = reserved
TMASKT15 = addr of local terminal NETNAME table (TCNT) entries
TMASKT16 = addr of autoinstall terminal model (AITM)
           table entries
TMASKT17 = addr of signon table (SNT) entries
TMASKT18 = addr of session (TCSE) entries
TMASKT19 = addr of remote connection entries (TCSR)*
TMASKT20 = addr of indirect connection entries (TCSI)*
TMASKT21 = addr of connection NETNAME (TCSN) entries*
TMASKT22 = addr of remote terminal entries (TCTR)*
TMASKT23 = addr of generic connection NETNAME (TCSM) entries*
TMASKT24 = addr of remote terminal NETNAME (TCNR) entries*

* - Secondary index
```

   Use the following formula to find the offset of the individual scatter table:

```
Length(TMATTV) * (n-1) + X'08'
```

   Where n = position in table (see above - TMASKTn)

To find Length(TMATTV) (and the value of n) see the *CICS Data Areas* manual.

6. Find the first directory element from its address in field SKTFDEA (offset X'10') in the hash table area.
7. Directory elements are chained together in alphabetic order. The address of the next element is in field DIRGNCHN (offset X'10').
8. Look at each directory element until you find the name of the control block you are looking for. The name is in field DIRKEY (offset X'18'). Field DIRTEA (offset X'0') holds the address of the desired control block.

Control blocks

Figure 76 shows the table manager control blocks.

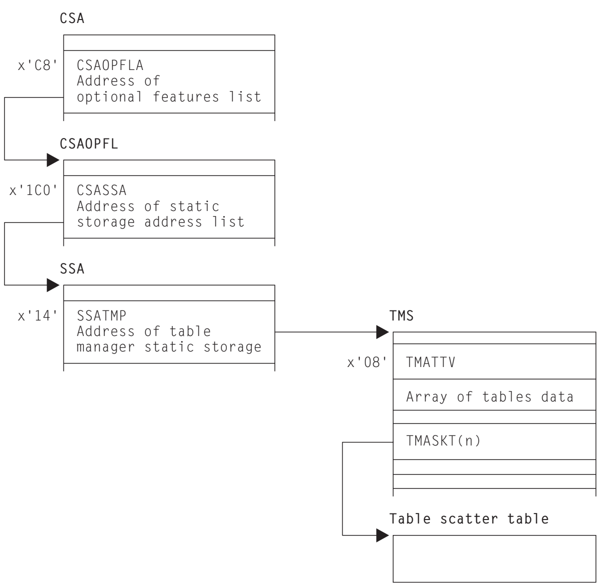


Figure 76. Table manager control blocks

See the *CICS Data Areas* manual for a detailed description of these control blocks.

Modules

DFHTMP

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for this function:
- AP 00EA, for which the trace level is AP 1.

See the *CICS Trace Entries* for further information.

Statistics

The statistics utility program, DFHSTUP, provides, for table management, statistics (for each table) on the amount of storage (expressed in bytes) used by the table manager to support each table (excluding storage used for the tables themselves).

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 53. Task-related user exit control

Task-related user exit support in CICS, also known as the resource manager interface (RMI), provides an interface that non-CICS resource managers can use to communicate with CICS applications. The exit program can be enabled or disabled dynamically, and useful information can be transferred to a user work area.

Functional overview

The following operations may be performed on a task-related user exit from application programs:

ENABLE
: This is a global operation that names the task-related user exit and causes the task-related user exit to be loaded into storage, if it has not already been loaded. It also causes the exit program control block (EPB), which represents the task-related user exit, and the exit's global storage to be set up by the user exit manager module, DFHUEM. The EPB also holds a TALENGTH argument and a bit-string profile for use in an exit operation. The ENABLE operation does not pass control to the task-related user exit. DFHUEM is used to enable both global user exits and task-related user exits.

  The ENABLE operation is performed in two stages:
  1. ENABLE
  2. START.

  An exit is not made available for execution until it has been both enabled and started.

  You can use the TASKSTART keyword on the ENABLE command to enable a task-related user exit so that it is invoked at task start for all tasks in the CICS system.

  You can also enable a task-related user exit with the FORMATEDF keyword, which means that the task-related user exit can provide formatted screens for EDF to display, whenever a DFHRMCAL request to the task-related user exit takes place.

  The task-related user exit is invoked in the addressing mode of its original caller unless the LINKEDITMODE keyword is specified on the ENABLE command, in which case the exit is invoked in its own link-edit AMODE. LINKEDITMODE is only valid on the first ENABLE command for an exit program.

EXTRACT
: Information concerning an "enabled and started" task-related user exit is returned to an application when it issues this command.

DISABLE
: This is a global operation which in general terms is the reverse of an ENABLE request. The DISABLE operation can be performed in two stages:
  1. STOP: This is the reverse of the START keyword on the ENABLE request. It causes the task-related user exit to remain in main storage together with all its associated control blocks; however it is not available for execution until an ENABLE command with the START option is specified.
  2. EXITALL: This causes the EXIT and its control blocks to be deleted from main storage. The EPB however is added to a chain of re-usable EPB's anchored in the UETH. This function should not be used until all tasks that have used the exit have ended; the results of EXITALL before that point are unpredictable.

DFHRMCAL
: After an exit has been enabled and started, it can be invoked from an

application using a DFHRMCAL request directly, or by passing control to a stub which performs the DFHRMCAL request. A register 1 parameter list may be supplied to the task-related user exit from the application.

The task interface element (TIE) control block is created for the task and task-related user exit combination when the task issues its first DFHRMCAL request, unless the TIE has already been created because the task-related user exit was enabled for TASKSTART.

When a DFHRMCAL request is issued, control passes to DFHEIP, to DFHERM (the external resource manager interface program), and then to the task-related user exit. DFHERM manages the TIEs.

ENABLE, DISABLE, and EXTRACT are all EXEC CICS requests. DFHRMCAL is a macro.

A task-related user exit can "express interest" in certain types of events, and be invoked when these events take place. These events are:

- Application invocations (DFHRMCAL mentioned above), associated with which are optionally the EDF screen format invocations
- System Programming interface events i.e. INQUIRE EXITPROGRAM commands
- Syncpoint related events
- Task termination events
- CICS termination.

By default, it is assumed that task-related user exits are interested in application invocations only.

Design overview

The task-related user exit interface is comparable with the EXEC interface. When an application program requests the services of a non-CICS resource manager, it does so by a module called the task-related user exit. The exit receives arguments from the application program, and passes them on to the resource manager in a suitable form.

The advantage of this method is that if the resource manager is changed, the application program that invokes the resource manager should not need to be changed too.

The exit is part of the resource manager programs. The name of the exit, or the name of the entry to the exit, is specified by the resource manager, and each application program that invokes the resource manager has to be link-edited with an application program stub that refers to that name.

The exit is enabled and disabled using the user exit manager (DFHUEM). For enabling, the resource manager can specify the size of a task-related work area that it requires.

The exit, when enabled and subsequently driven, receives arguments in the form specified by the DFHUEXIT TYPE=RM parameter list (see the *CICS Customization Guide* or the manual). Register 1 points to this parameter list. Register 13 points to the address of a save area, rather than the address of the CSA. The save area is 18 words long, with registers 14 through 12 stored in the fourth word onward.

Responses to the request are indicated by values placed in register 15, and also by means that are specific to the architecture of the application interface, for example, by moving data into storage areas passed by the call, or into the caller's register 15.

The main control blocks used by the task-related interface are the task interface element (TIE):

- A TIE is created by DFHERM on the first call by a task to each resource manager, and it is chained to the TCA for that task.

Task-related user exit implementation

The state of an exit is managed by DFHUEM, which is described under Chapter 64, “User exit control,” on page 467. For an exit, the TALENGTH argument and a profile in the form of a bit-string are held in the exit program block (EPB). These arguments are not processed until the occurrence of an application program CALL that explicitly names the exit, unless the TASKSTART keyword is used on the ENABLE request.

Entry to the exit is through the task-related user exit interface, which comprises:

- An application stub provided with the exit, but generated using the CICS-provided macro DFHRMCAL. It is this stub which explicitly names the exit, and which is link-edited with each application program that uses the application program interface (API) of the resource manager.
- DFHEIP, which is entered at DFHEIPCN by the application stub, in much the same way as EXEC CICS commands are routed at execution time.
- DFHERM, which receives control when DFHEIP discovers that the call is not for a CICS control function, but for a named exit.

DFHERM receives a set of registers (those of the caller, for example, the application program), and a routing argument which names the exit. This routing argument is constructed by DFHRMCAL, in the application stub, and is not normally visible to the application programmer. DFHERM retrieves the name of the requested exit from the routing argument, and scans any existing task interface elements (TIEs) that are chained from the task’s TCA, looking for a TIE associated with the named exit. If such a TIE is not found, it searches the installed exits on a chain of EPBs, looking for the matching name. On finding a match, DFHERM constructs a TIE to represent the connection between that task and the exit. The TIE is initialized from information provided in the EPB; the TALENGTH argument defines the size of a task-local work area which can be thought of as a logical extension of the TIE. The profile string is also copied into the TIE.

DFHERM stacks (stores in a last-in, first-out manner) various parts of the program execution environment—the status of HANDLE commands, file browse cursors, the EXEC interface block (EIB), and so on—and builds a parameter structure which is essentially a superset of that built by DFHUEH. Additional arguments include the task-local work area, the profile referred to above, and an 8-byte UOW identifier supplied by Recovery Manager.

DFHERM then passes control to the exit’s entry point using standard CALL conventions, in which register 13 addresses a save area for DFHERM’s own registers, register 14 addresses DFHERM’s next sequential instruction, and register 1 addresses the passed parameters. This is a vector of addresses which include that of the caller’s register save area. Any changes the exit makes to arguments of the application program interface (API), or to the contents of the caller’s register save area, are not examined by DFHERM when it regains control, because they are not part of the CICS task-related user exit interface—rather they are the concern of the caller and the exit. However, the exit can request DFHERM to schedule certain actions by means of the profile argument. For example, the exit can request that it be informed (driven) when commitment of resources (syncpointing) is taking place, or the exit can request that DFHERM no longer routes API calls to it from this task.

Finally, on regaining control from the exit, DFHERM unstacks the objects that it had previously stacked, and returns to the caller. The state of the cursors, HANDLE labels, and so on, is apparently unchanged by the actions of DFHERM or the exit. Note that the exit may have used EXEC CICS HANDLE commands; this does not interfere with the caller’s HANDLE status.

In the discussion of DFHERM so far, the term “caller” has been used for the application program. However, a caller can be a function such as syncpoint (DFHERMSP), task control (DFHAPXM or DFHERMSP), system programming interface (DFHUEIQ), CICS termination (DFHAPDM or DFHSTP) or EDF (DFHERM). The exit can set appropriate bits in the profile (schedule flag word) so that, if the corresponding function is subsequently invoked, it in turn calls the exit. The exit can determine the identity of the caller from the first argument (called the “function definition”). This argument, passed by DFHERM, always has its first byte equal to X'00'. (If the first byte is other than X'00', the exit has been entered from

DFHUEH as a global user exit.) DFHERM sets the second byte of this argument according to the type of caller, thus indicating which interface is addressed by the caller's register save area. The second byte is:

X'01' For system programming interface

X'02' For an application program

X'04' For the syncpoint program

X'08' For CICS task control

X'0A' For a CICS termination call

X'0C' For an EDF call.

Any remaining arguments are specific to each individual caller.

The flow of control for the task-related user exit interface is shown in Figure 77.

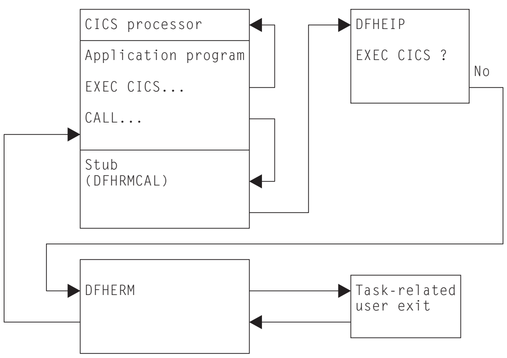


Figure 77. Task-related user exit control flow

Processors

The term "processor" is used to refer to two different types of object:

1. For the EXEC interface, it refers to the function-dependent modules associated with the EXEC interface nucleus, DFHEIP. These processors usually have names such as DFHEPC, DFHETC, DFHETD, and so on, and each of these is invoked by DFHEIP. DFHERM is also a processor of this type.
2. In various contexts, including task-related user exits, it refers to a piece of code that is link-edited with an application program and serves the dual function of:
   - Satisfying the CALL requirement for a target address—its entry resolves a V-type ADCON
   - Finding the entry point of DFHEIP.

Both these types of processor are part of the path between an application call and the functional control module that supports the request. This path appears as follows:

```
Application call
 Application processor (type 2)
  DFHEIP
   EXEC interface processor (type 1)
    Functional control module
```

Examples of the interface are:

```
EXEC CICS SYNCPOINT ... CICS API
 DFHECI     CICS COBOL EIP router
  DFHEIP
   DFHEISP  CICS syncpoint router
    DFHSPP  CICS syncpoint manager
```

```
       CICS Recovery manager domain

EXEC DLI TERM ... DLI HLPI
 DFHECI     CICS COBOL EIP router
  DFHEIP
   DFHERM   CICS RMI module
    DFHEDP  DLI HLPI manager
             (implemented as a task-related
                user exit)
```

Control blocks

The control blocks used in task-related user exit control are the exit program control block (DFHEPB), the task interface element (DFHTIEDS).

Figure 78 shows the main control blocks associated with task-related user exits.

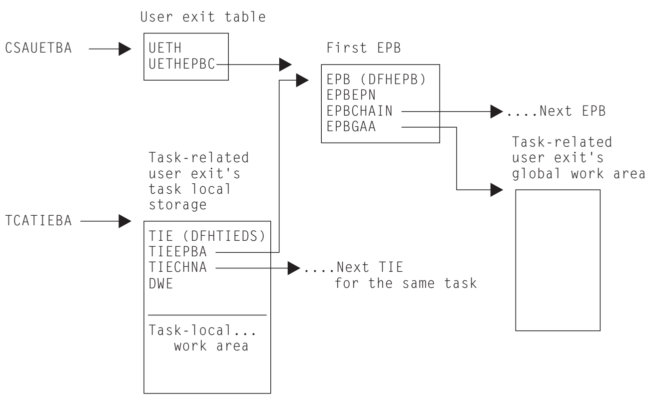


Figure 78. Control blocks associated with task-related user exits

Field CSAUETBA in the CSA points to the user exit table (UET); UETHEPBC in the UET points to the first exit program block (EPB); and EPBCHAIN in each EPB points to the next EPB in the chain.

Each EPB holds:
- The address of the exit's entry point (EPBEPN)
- The address of the global work area
- The halfword length of the global work area
- The halfword length of the task-local work area.

One EPB is associated with each enabled task-related user exit program or entry name.

EPBs used for global user exits and for task-related user exits are held on the same EPB chain.

The task-related user exit's global storage is optional. It is associated with an individual enabled task-related user exit program or entry name. Several task-related user exit programs or entry names can share the same global storage.

For full details of the EPB, see the *CICS Data Areas* manual.

The task interface element (TIE) is associated with each associated pair of CICS task and task-related user exit. The first time a CICS task passes control to a particular task-related user exit, a TIE is created. The TIE lasts until task termination.

Task-related user exit control

Note that all TIEs relating to a single task are chained together (more than one TIE is set up when a single CICS task makes use of more than one task-related user exit). The TIEs corresponding to a single EPB (that is, to a single task-related user exit program or entry name) are not chained together.

A global user exit may only use global storage; a task-related user exit may use both global storage and task-local work area.

Field TCATIEBA in the TCA points to the first TIE, and TIECHNA in each TIE points to the next TIE in the chain.

The TIE holds information relevant to all invocations of the task-related user exit for the task concerned. For example, TIEFLAGS holds information concerning the events for which the task-related user exit should be invoked, for example, API calls, syncpoint, and task start.

Figure 79 gives a closer look at the TIE control block chain that is used during the lifetime of a task-related user exit.

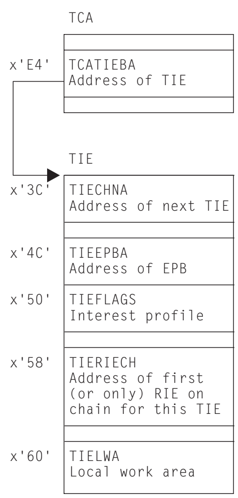


Figure 79. Control blocks used during the lifetime of a task-related user exit

For full details of the TIE control blocks, see the *CICS Data Areas* manual.

Modules

Module	Function
DFHUEM	The EXEC interface processor for the ENABLE, DISABLE, and EXTRACT user exit commands.
DFHERM	Interfaces with task-related user exit.
DFHTIEM	Handles the TIE subpools.

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for this function:
- AP 2520) for which the trace level is RI 1.
- AP 2521)
- AP 2522) for which the trace level is RI 2.
- AP 2523)

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

External interfaces

Calls are made to the task-related user exit via DFHEIP and DFHERM from the following modules:

DFHAPXM	Task start
DFHERMSP	Task end
DFHERMSP	Syncpoint and backout
DFHRMSY	For syncpoint resynchronization
DFHAPDM	CICS termination
DFHSTP	CICS termination
DFHUEIQ	System programming interface for inquire exitprogram calls
Applications	Application calls to resource manager
DFHERM	EDF invocations for application calls to resource manager

Chapter 54. Task-related user exit recovery

Task-related user exit recovery, also known as the resource manager interface (RMI) recovery, ensures that changes to recoverable resources performed by an external resource manager in a logical unit of work are either all committed or all backed out.

Design overview

During the execution of a CICS task, the CICS recovery manager communicates with the resource manager task-related user exit to prepare to commit, to commit unconditionally, or to back out. The purpose of these calls is to ensure that changes to recoverable resources performed in a unit of work (UOW) are either all committed or all backed out, if there is a failure anywhere in CICS or in any of the external resource managers.

Each UOW created by Recovery Manager Domain is identified by a UOW_ID and a Local UOW_ID. The LOCAL UOWID is an eight byte value whose format is easy for CICS to identify whether the UOW originated before or after an initial start.

When the resource manager receives the call to commit unconditionally or to back out, it takes the corresponding irreversible step, if possible. If the action is successful, the resource manager sends the appropriate return code. If not, it sends a return code which requests that CICS record the state of the UOW, and tries to resolve the status at a later time.

Recovery manager domain maintains the status of UOWs that require resynchronization, until all participants in the UOW have successfully resynchronized. Recovery manager domain maintains these UOWs across cold, warm and emergency start of CICS. An initial start of CICS however will mean that Recovery manager domain will lose this information and resynchronization will not be possible.

The RMI also supports an optimized syncpoint process to improve performance under certain conditions where a single-phase commit can be used. With single phase commit Recovery manager does not have to maintain resynchronization information for the RMI. This optimized process is described in more detail later in this section.

The two-phase commit process

The RMI supports the two-phase commit process. The following is a brief summary of the two-phase commit process and other related processing as seen from the RMI's point of view.

- When a unit of work is first created, Recovery manager creates local_uow_id which will be used by the RMI.
- When the task syncpoints, a prepare-to-commit request is then issued to each task-related user exit used during the current UOW. For each task-related user exit, issuing the prepare request indicates the start of phase 1 of commit processing from CICS's point of view.
- If all syncpoint participants vote 'YES' to the prepare requests, then Recovery manager will commit the UOW. CICS then invokes each task-related user exit with a commit request. This indicates the start of phase 2 commit processing for the task-related user exit.

  If the task-related user exit is unable to commit the UOW, Recovery manager will maintain a record of the UOW's status so that the task related user exit can resync later.
- If one or more of the task-related user exits votes 'NO' to the prepare-to-commit request, all the task's recoverable resources are backed out.

Resolution of in-doubts

An external resource manager can be left in doubt about the disposition of UOWs, for example, if the resource manager abnormally terminated after receiving a prepare request for an UOW, but before receiving the commit or backout request. The resource manager, at any time while interfacing with CICS, can supply a list of recovery tokens representing the in-doubt UOWs to the task-related user exit. The task-related user exit (or other related code) can then issue an EXEC CICS RESYNC request with the in-doubt list and the name of the task-related user exit as parameters.

As a result of a the EXEC CICS RESYNC command, DFHERMRS initiates a CRSY task (running program DFHRMSY) for each UOW named in the indoubt list passed from the TRUE. DFHRMSY interfaces with Recovery manager to find out the status of the UOW, and calls the task-related user exit with the appropriate resolution, for example 'Commit', 'Backout' and so on. For each successful commit or backout, DFHRMSY informs Recovery manager that it can delete the TRUEs involvement in the UOW. When all interested parties in a UOW complete such processing, Recovery manager deletes its record of the UOW.

If an EXEC CICS RESYNC request is issued without an in-doubt list or with an in-doubt list of length zero, then DFHERMRS informs Recovery manager that it can remove the TRUE (identified by its name and qualifier) from all UOWs in the resynchronization set, i.e. delete all resync information for a TRUE.

A resynchronization set is first established when a TRUE is enabled. The next resynchronization set is identified on completion of an EXEC CICS RESYNC command, and is used for the next RESYNC command. A resynchronization bounds how many UOWs resync information is deleted for because RESUNC commands execute at the same time as new work is processed by a TRUE. A RESYNC command with a zero list should not delete resync information new UOW created since the resync command was issued.

The single-phase commit process

The RMI also supports the single-phase commit process for UOWs that are read-only, and for UOWs where CICS detects that only one external resource manager has been called for update requests. The task-related exit must indicate to the RMI that it is capable of processing single-phase commit requests; otherwise, a two-phase commit is used. Use of single-phase commit improves performance, because CICS does not perform any logging and the task-related user exit is called only once during syncpoint processing.

Single-phase commit for read-only UOWs

To take advantage of single-phase commit for read-only UOWs, the external resource manager must return to the task-related user exit an indicator that the UOW is read-only. This can be done by the resource manager returning a flag indicating the “history” of the UOW so far (that is, whether it is read-only so far), or returning information about the current request. In the latter case, it is the responsibility of the task-related user exit to keep a “history” of the UOW so far. After each request, the task-related user exit must return to CICS with a flag set in the parameter list indicating this history.

At syncpoint time, if CICS detects that the UOW is read-only, it invokes the task-related user exit with an “End-UOW” request instead of the normal prepare and commit requests associated with a two-phase commit. This means that the task-related user exit is invoked only once during syncpoint. The “End-UOW” request is issued during phase 2 syncpoint processing. On receiving an “End-UOW” request, the task-related user exit should invoke the resource manager for single-phase commit. There are no return codes associated with the “End-UOW” request, and CICS does not perform any logging for this type of request.

Single-phase commit for the single updater

To take advantage of single-phase commit for the single-update situation, the task-related user exit must indicate to the RMI that it knows the single-update protocol. It does this by setting a flag in the parameter list at the same time as it expresses an interest in syncpoint.

At syncpoint time, if CICS detects that only resources owned by one external resource manager were updated in the UOW, and if the task-related user exit has indicated that it understands the protocol, CICS invokes the task-related user exit with an ‘Only’ request, instead of the normal prepare and commit requests associated with a two-phase commit. This means that the task-related user exit is invoked only once during syncpoint. The ‘Only’ request is issued during phase 1 syncpoint processing. CICS does not perform any logging for this type of request. When invoked for an ‘Only’ request, the task-related user exit should invoke the resource manager for single-phase commit.

There are two architected responses to the ‘Only’ request: ‘OK’ and ‘Backed-out’. ‘OK’ means that the UOW was committed; ‘Backed-out’ means that the single-phase commit failed and the updates were backed out. It is important to note that, unlike the two-phase commit, there is no equivalent ‘Remember’ response. If a task-related user exit calls a resource manager for single-phase commit and, for example, the resource manager abends while processing this request, the task-related user exit is left in doubt as to the outcome of the request. The task-related user exit cannot return to CICS in this case, but instead must output diagnostic messages as appropriate, and then abend the transaction.

Recovery manager does not keep resynchronization information for UOWs using single phase commit. Because the resource manager is the only updater in the UOW, CICS is ***not*** in doubt about any of its resources. The resource manager has either committed or backed out the updates. The messages output by the task-related user exit, in conjunction with any messages output by the resource manager, can be used to determine the outcome of the UOW.

Modules

Module	Function
DFHERMRS	DFHERMRS is invoked by DFHEISP as a result of a an EXEC CICS RESYNC command. It attaches a CRSY task for each UOW identified in the IDLIST. Calls Recovery manager to delete unwanted resynchronization information.
DFHRMSY	A CRSY task (running program DFHRMSY) is attached for each indoubt UOW appearing in the in-doubt list for an EXEC CICS RESYNC command. This program then issues the appropriate ‘phase 2 of syncpoint’ request, that is, commit or backout, to the external resource manager that issued the EXEC CICS RESYNC.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:
- AP 2540) For trace level RI Level 1
- AP 2541)
- AP 2548) For trace level RI level 2
- AP 2549)
- AP 2560) For trace level RI level 1
- AP 2561)

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

External interfaces

Calls are made from DFHRMSY, via DFHERM, to the task-related user exit to provide information about the disposition of the UOW, when resynchronization of in-doubts is taking place.

Chapter 55. Terminal abnormal condition program

| Terminal error processing for BSAM-supported terminals normally routes any error to the terminal abnormal condition program (DFHTACP). Depending on the type of error, DFHTACP issues messages, sets error flags, and places the terminal or line out of service.

Before default actions are taken, CICS passes control to the terminal error program (DFHTEP) for application-dependent action if necessary. On return from the terminal error program, DFHTACP performs the indicated action as previously set by DFHTACP or as altered by the TEP, a sample version of which is supplied by CICS (DFHXTEP in source code form). See Chapter 57, “Terminal error program,” on page 425 for further information about the TEP.

Design overview

The terminal abnormal condition program (DFHTACP) is used by terminal control to analyze any abnormal conditions. Appropriate action is taken with regard to terminal statistics, line statistics, terminal status, and line status; the task (transaction) can be terminated. Messages are logged to the transient data master terminal destination (CSMT) or the terminal log destination (CSTL). DFHTACP links to the user-supplied (or sample) terminal error program, passing a parameter list via a COMMAREA that is mapped by the DFHTEPCA DSECT. This allows the user to attempt recovery from transmission errors and to take appropriate action for the task.

Table 24 lists the various TACP message processing routines, which assemble the text of the messages and write them to one of three destinations depending on the type of error.

The matrix shown in Table 25 on page 402 shows the the sequence in which the message routines are called for each error code. For example, for error code X'88', the processing routines are executed in the following order: ME, F, W, X, N, BA, and finally R.

Table 26 on page 403 gives a generalization of TACP’s default error handling upon completion of the message processing. For each error code, it shows the first routine to be called.

Table 24. TACP message routines

Routine	Function
A	Establish DFHTC message number 2501 (Msg too long, please resubmit)
D	Establish DFHTC message number 2502 (TCT search error)
F	Establish DFHTC message number 2507 (Input event rejected)
H	Establish DFHTC message number 2506 (Output event rejected)
I	Establish DFHTC message number 2513 (Output length zero)
J	Establish DFHTC message number 2514 (No output area provided)
K	Establish DFHTC message number 2515 (Output area exceeded)
L	Establish DFHTC message number 2517 (Unit check SNS=ss, S.N.O.)
M	Establish DFHTC message number 2519 (Unit exception, S.N.O.)
N	Generate standard message inserts, for example, ‘at term tttt’
O	Generate special inserts for message DFHTC2500
Q	Write to terminal causing the error, after retrieving the message text from ME domain using an MEME RETRIEVE_MESSAGE call
R	Write to destination (CSMT or CSTL) using an MEME SEND_MESSAGE call to ME domain
T	Obtain terminal main storage area (message build area)

Table 24. TACP message routines (continued)

Routine	Function
V	Establish DFHTC message number 2511 (Incorrect write request)
W	Establish 'return code xx' message insert
X	Convert hexadecimal byte into 2 printable characters
AB	Establish DFHTC message number 2534 (Incorrect destination)
AE	Establish DFHTC message number 2500 (Line|CU|Terminal out of service)
AF	Obtain terminal statistics
BA	Obtain line statistics
BB	Establish DFHTC message number 2516 (Unit check SNS=ss)
BC	Establish DFHTC message number 2518 (Unit exception)
BF	Establish DFHTC message number 2521 (Undetermined unit error)
CA	Establish DFHTC message number 2522 (Intercept required)
DB	Establish DFHTC message number 2529 (Unsolicited input)
ME	Initialize parameter list for calling ME domain

Table 25. TACP message construction matrix

Error codes																
X'81'	**X'82'**	**X'84'**	**X'85'**	**X'87'**	**X'88'**	**X'8C'**	**X'8D'**	**X'8E'**	**X'8F'**	**X'94'**	**X'95'**	**X'96'**	**X'97'**	**X'99'**	**X'9A'**	**X'9F'**
ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME
T																
	AE															
		D														
			V													
				DB												
					F											
						H										
							I									
								J								
									K							
										BB						
											L					
												BC				
													M			
														BF		
															CA	
																AB
A																
	O															
					W	W										
AF																
Q																
					X	X				X	X					
		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
		BA			BA	BA										

Table 25. TACP message construction matrix (continued)

Error codes																
X'81'	X'82'	X'84'	X'85'	X'87'	X'88'	X'8C'	X'8D'	X'8E'	X'8F'	X'94'	X'95'	X'96'	X'97'	X'99'	X'9A'	X'9F'
	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 26. TACP default error handling

Error code	Default action
X'81'	Abend transaction
X'82'	none
X'84'	Put line in or out of service, as required
X'85'	Abend transaction
X'87'	Unsolicited input message
X'88'	Put line (or terminal) out of service
X'8C'	Put line (or terminal) out of service
X'8D'	Abend transaction
X'8E'	Abend transaction
X'8F'	Abend transaction
X'94'	I/O error test
X'95'	I/O error test
X'96'	I/O error test
X'97'	I/O error test
X'99'	Put line (or terminal) out of service
X'9A'	Test line for next operation
X'9F'	Abend transaction

Modules

DFHTACP

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for the terminal abnormal condition program:

- AP 00E6, for which the trace level is TC 1.

DFHTACP provides trace entries immediately before and after calling DFHTEP.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 56. Terminal control

Terminal control allows communication between terminals and application programs. VTAM/NCP is used for most terminal data control and line control services.

Terminal control supports automatic task initiation to process transactions that use a terminal but which are not directly initiated by the terminal operator (for example, printers).

Terminal control can also provide a simulation of terminals, using sequential devices, in order to help test new applications.

Design overview

The user can specify that concurrent terminal support is to be provided by any combination of the following access methods:
- VTAM
- Basic sequential access method (BSAM)
- Interregion communication (IRC)
- Console.

The primary function of terminal control is to take an input/output (I/O) request for a terminal and convert it to a format acceptable to the access method (VTAM or BSAM).

Terminal control uses data that describes the communication lines and terminals, kept in the terminal control table (TCT). The TCT is generated by the user as part of CICS system definition, or dynamically as needed. The TCT entries contain terminal request indicators, status, statistics, identification, and addresses of I/O and related areas.

When CICS terminal control is used with VTAM, VTAM itself resides in a separate address space, having a higher priority than CICS. VTAM-related control blocks and support programming comprise the CICS terminal control component. The application programs that run under CICS control communicate with terminals through the CICS terminal control interface with VTAM.

VTAM network functions allow terminals to be connected to any compatible control subsystem that is online. This enables a terminal operator to switch from one CICS system to another, or to another subsystem.

VTAM manages the flow of data between devices in the network and VTAM application programs such as CICS. VTAM is responsible for:
- Connecting, controlling, and terminating communication between the VTAM applications and terminal logical units
- Transferring data between VTAM applications and logical units
- Allowing VTAM applications to share communication lines, communication controllers, and terminals
- Controlling locally attached devices, that is, those not connected through a communication controller
- Providing tools to monitor network operations and make dynamic changes to the network configuration.

In a VTAM environment, the functions of CICS terminal control include:
- Establishing communication with terminal logical units (LUs) by issuing logon requests, communicated through the access method
- Handling terminal input and passing user program requests for communication to VTAM
- Returning terminal LUs to the access method by accepting logoff requests
- Taking measures to ensure the integrity of messages flowing to and from VTAM

- Performing logical error recovery processing for VTAM devices.

Terminal control issues VTAM macros to receive incoming messages, and routes them to the appropriate CICS application program for processing. Likewise, it sends messages destined for various devices in the network to VTAM, which then routes them to the appropriate location.

Terminal control services

The following services are performed by, or in conjunction with, terminal control:
- Service request facilities
- System control services
- Transmission facilities.

Service request facilities

Write request
: Sets up and issues or queues access method macros; performs journaling and journal synchronization.

Read request
: Sets up and issues access method macros; performs journaling if required.

Wait request
: Causes a dispatcher to suspend.

Dispatch analysis
: Determines the type of access method and terminal used, and executes the appropriate area of terminal control.

System control services

Automatic task initiation
: Services requests for automatic task (transaction) initiation caused by events internal to the processing of CICS.

Task initiation
: Requests the initiation of a task to process a transaction from a terminal. When an initial input message is accepted, a task is created to do the processing.

Terminal storage
: Performs allocation and deallocation of terminal storage.

Transmission facilities—VTAM

Connection services
: Accepts logon requests, requests connection of terminals for automatic task initiation, and returns terminals to VTAM, as specified by the user. If the terminal has not been defined, CICS uses the VTAM logon information to autoinstall the terminal.

Transmission facilities—VTAM/non-VTAM

Access method selection
: Passes control to the appropriate access method routine based on the access method specified in the terminal control table.

Wait
: Synchronizes the terminal control task with all other tasks in the system. When all possible read and write operations have been initiated, terminal control processing is complete and control is returned to the transaction manager to allow dispatching of other tasks.

Terminal error recovery

The resolution of certain conditions (for example, permanent transmission errors) involves both CICS and additional user coding. CICS cannot arbitrarily take all action with regard to these errors. User application logic is sometimes necessary to resolve the problem.

For the VTAM part of the network, terminal error handling is carried out by the node abnormal condition program (NACP) and a sample node error program (NEP), provided by CICS, or a user-written node error program. For further information about these, see Chapter 36, “Node abnormal condition program,” on page 327 and Chapter 37, “Node error program,” on page 331.

For the portion of the telecommunication network connected to BSAM, these error-handling services are provided by the terminal abnormal condition program (TACP) and by the user-written or sample terminal error program (TEP). For further information about these, see Chapter 55, “Terminal abnormal condition program,” on page 401 and Chapter 57, “Terminal error program,” on page 425.

The following sequence of events takes place when a permanent error occurs for a terminal:

1. The terminal is “locked” against use.
2. The node or terminal abnormal condition program is attached to the system to run as a separate CICS task.
3. The node or terminal abnormal condition program writes the error data to a destination in transient data control if the user has defined one. This destination is defined by the user and can be intrapartition or extrapartition.
4. The node or terminal abnormal condition program then links to the appropriate node/terminal error program to allow terminal- or transaction-oriented analysis of the error. In the node or terminal error program, the user may decide, for example, to have the terminal placed out of service, have the line placed in or out of service, or have the transaction in process on the terminal abnormally terminated.
5. The terminal is “unlocked” for use.
6. The node or terminal abnormal condition program is detached from the system if no other terminals are to be processed.

Testing facility—BSAM

To allow the user to test programs, BSAM can be used to control sequential devices, such as card readers, printers, magnetic tape, and direct-access storage devices. These sequential devices can then be used to supply input/output to CICS before actual terminals are available or during testing of new applications.

Terminal control modules (DFHZCP, DFHTCP)

Terminal control consists of two CICS resource managers:

ZCP
: DFHZCP, DFHZCX, and DFHZCXR provide both the common (VTAM and non-VTAM) interface, and DFHZCA, DFHZCB, DFHZCC, DFHZCW, DFHZCY, and DFHZCZ provide the VTAM-only support.

TCP
: DFHTCP provides the non-VTAM support (not MVS console support).

Terminal control communicates with application programs, CICS system control functions (transaction manager, storage control), CICS application services (basic mapping support and data interchange program), system reliability functions (abnormal condition handling), and operating system access methods (VTAM or BSAM).

Requests for terminal control functions made by application programs, BMS, or the transaction manager, are processed through the common interface of DFHZCP. Generally, terminal control requests for other CICS or operating system functions are issued by either ZCP or TCP, depending upon the terminal being serviced.

Terminal control

The ZCP and TCP suites of programs are loaded at CICS system initialization according to specified system initialization parameters, as follows:

- DFHTCP is loaded only if TCP=YES is specified.
- DFHZCP, DFHZCX, and DFHZCXR are always loaded.
- DFHZCA, DFHZCB, DFHZCY, and DFHZCZ are loaded only if VTAM=YES is specified.
- DFHZCC and DFHZCW are loaded only if ISC=YES is specified.

Figure 80 shows the relationships between the components of terminal control.

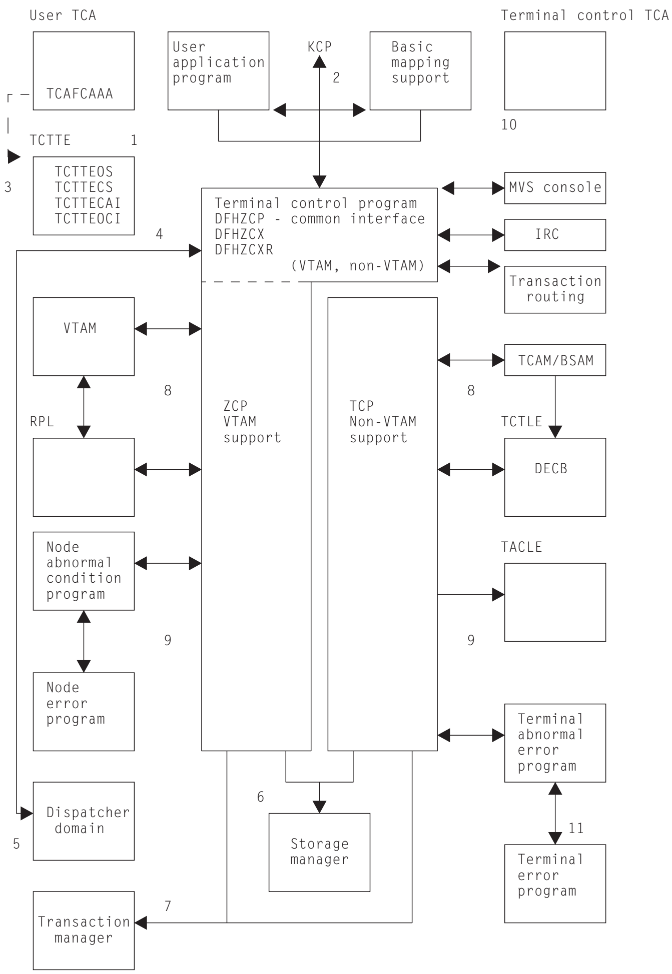


Figure 80. Terminal control interfaces

Notes for Figure 80:

Common interface

1. When a terminal control request is issued by an application program, or internally by the basic mapping support (BMS) routines using the DFHTC macro, request bits are set in the user's task control area (TCA) and control is passed to the common interface (VTAM, non-VTAM) routines of DFHZCP.
2. If the request includes WAIT and the IMMED option is not in effect, control is passed to the transaction manager to place the requesting program (task) in a suspended state. If WAIT is not included, control is returned to the requesting task.

3. The task's TCA contains the TCTTE address either in a field named TCAFCAAA (facility control area associated address) or in a field named TCATPTA when passing TCATPTA to terminal control.
4. The dispatcher dispatches terminal control through the common interface (DFHZDSP in DFHZCP) for one of the following reasons:
   - The system address space exit time interval (specified by the ICV system initialization parameter) has elapsed since the last terminal control dispatch.
   - The specified terminal scan delay (specified by the ICVTSD system initialization parameter) has elapsed.
   - There is high-performance option (HPO) work to process.
   - The terminal control event has been posted complete (for example, an exit scheduled in the case of VTAM, or an event control block (ECB) posted in the case of non-VTAM), and CICS is about to go into a wait condition.
5. Terminal control, through its common interface (DFHZDSP) requests the dispatcher to perform a CICS WAIT when the terminal control task has processed through the terminal network and has no further work that it can do.

Access method dependent interface

6. Terminal control communicates with storage manager to obtain and release storage as follows:

   VTAM
   ZCP modules issue domain calls for terminal storage (TIOAs), receive-any input area (RAIA) storage, and request parameter list (RPL) storage.

   Non-VTAM
   DFHTCP issues DFHSC macros to obtain and release terminal and line storage.
7. Terminal control communicates with the transaction manager by means of the DFHKC macro. The macro can be issued by certain CICS control modules, depending upon the terminal being serviced. Terminal control may request the transaction manager to perform one of the following:
   - Attach a task upon receipt of a transaction identifier from a terminal.
   - Respond to a DFHKC TYPE=AVAIL request (a task control macro documented only for system programming) when a terminal is required by or for a task and that facility is available.
8. Terminal control communicates with operating system access methods in either of the following ways, depending upon the terminal being serviced:

   VTAM
   ZCP (referring here to the resource manager) builds VTAM request information in the RPL which is then passed to VTAM for servicing. VTAM notifies terminal control of completion by placing completion information in the RPL. ZCP analyzes the contents of the RPL upon completion to determine the type of completion and the presence of error information. Communication with VTAM also occurs by VTAM scheduling exits, for example, LOGON or LOSTERM. VTAM passes parameter lists and does not always use an RPL.

   When authorized-path VTAM has been requested (HPO), communication with VTAM also occurs in service request block (SRB) mode (using DFHZHPRX); ZCP uses the RPL with an extension to communicate with its SRB mode code. When an SRB mode RPL request is complete, ZCP calls the relevant exit or posts the ECB, as indicated by the RPL extension.

   Non-VTAM
   DFHTCP builds access method requests in the data event control block (DECB), which is part of the terminal control table line entry (TCTLE). The DECB portion of the TCTLE is passed to the access method by terminal control to request a service of that access method. The access method notifies terminal control of the completion of the service through the DECB. Terminal control analyzes the contents of the DECB upon completion to determine the type of completion and to check for error information.

9. Terminal control communicates with the CICS abnormal condition functions in either of the following ways, depending upon the terminal being serviced:

   VTAM The activate scan routine (DFHZACT, in the DFHZCA load module) attaches the CSNE transaction to run the node abnormal condition program (DFHZNAC); this is done during CICS initialization. DFHZNAC does some preliminary processing and then passes control to the node error program (DFHZNEP). (The node error program can be either your own version or the default CICS-supplied version.) Upon the completion of the user's error processing, control is returned to DFHZNAC. (For further information about DFHZNAC, see Chapter 36, "Node abnormal condition program," on page 327.)

   Non-VTAM DFHTCP attaches the CSTE transaction to run the terminal abnormal condition program (TACP) and passes a terminal abnormal condition line entry (TACLE) when an error occurs. The TACLE is a copy of the DECB portion of the TCTLE and contains all information necessary for proper evaluation of the error, together with special action indicators that can be manipulated to alter the error correction procedure. After analyzing the DECB, DFHTACP calls the terminal error program (DFHTEP) with a COMMAREA containing the TACLE address. (The terminal error program can be either your own version or the default CICS-supplied version.) For further information about DFHTACP, see Chapter 55, "Terminal abnormal condition program," on page 401.

10. Terminal control is executed under either the user's TCA or its own TCA as follows:

    User's TCA

    a. During the application program interface
    b. During the interface with basic mapping support
    c. While performing direct VTAM terminal SEND requests.

    Terminal control's TCA

    a. When the dispatcher dispatches terminal control
    b. When terminal control issues a request to the transaction manager to attach a task
    c. When terminal control issues a request to storage control
    d. While performing non-VTAM terminal I/O or queued VTAM terminal I/O
    e. For session-control functions when no task is attached.

Because many devices are supported by CICS terminal control, a large number of modules are required to provide this support.

Figure 81 gives an overview of the relationships between the functions within terminal control and the rest of CICS and Figure 82 on page 411 through Figure 84 on page 413 show some of the flows through the terminal control modules.

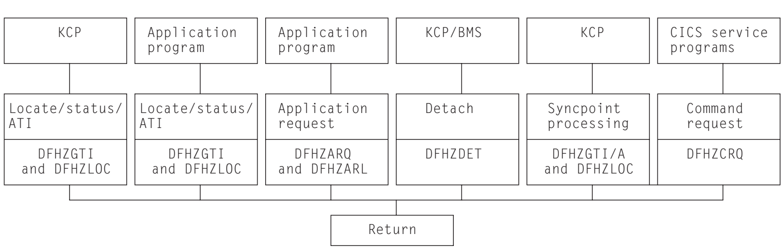


Figure 81. Terminal control functions and modules

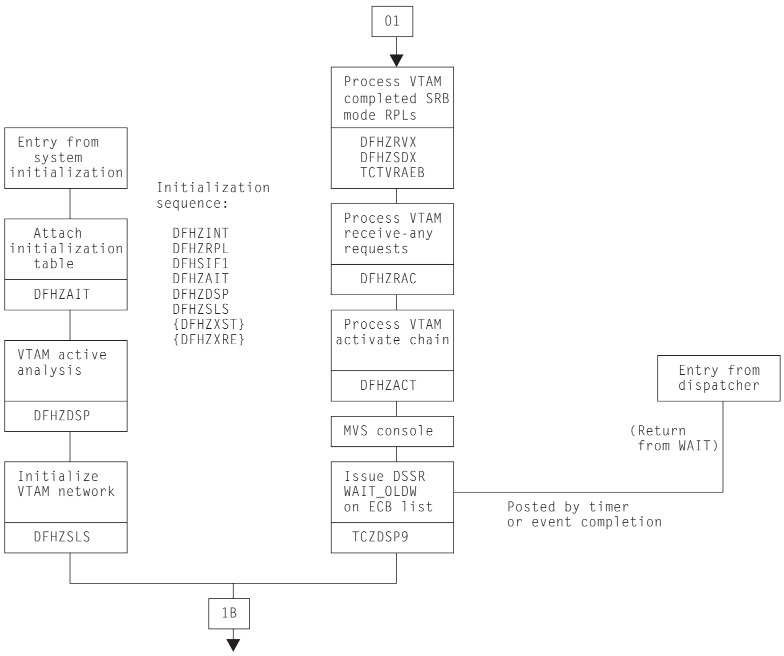


Figure 82. Terminal control ZCP and TCP common control routines

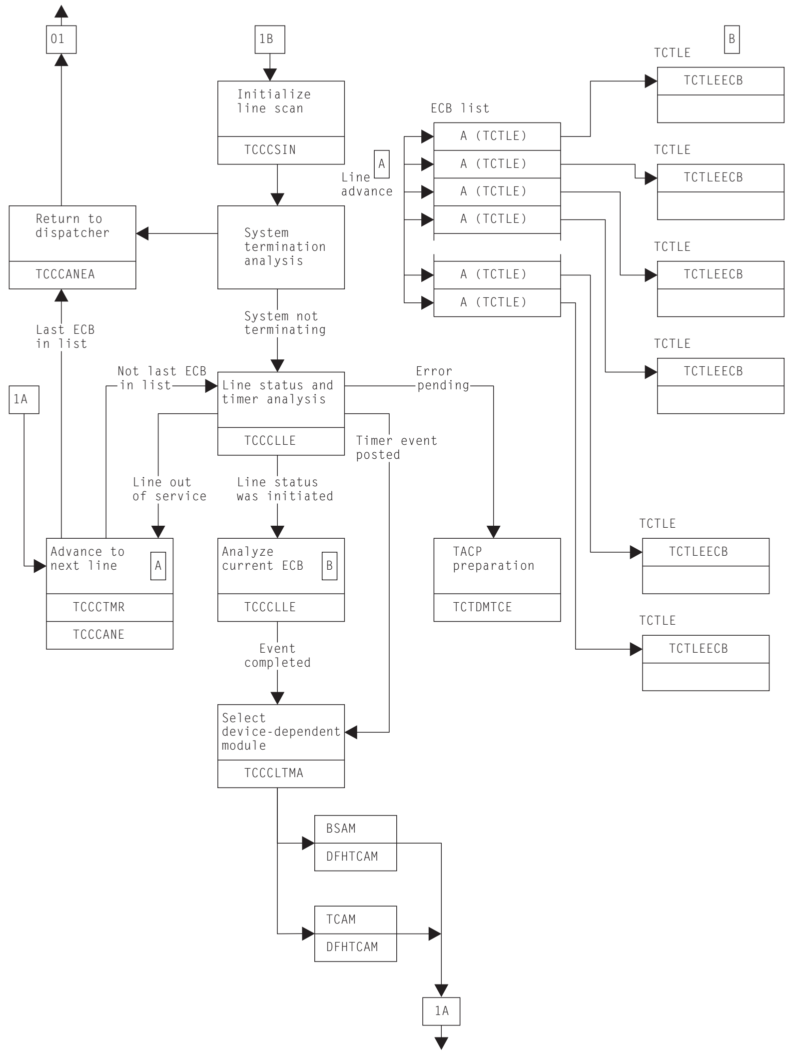


Figure 83. Terminal control TCP control routines (BSAM)

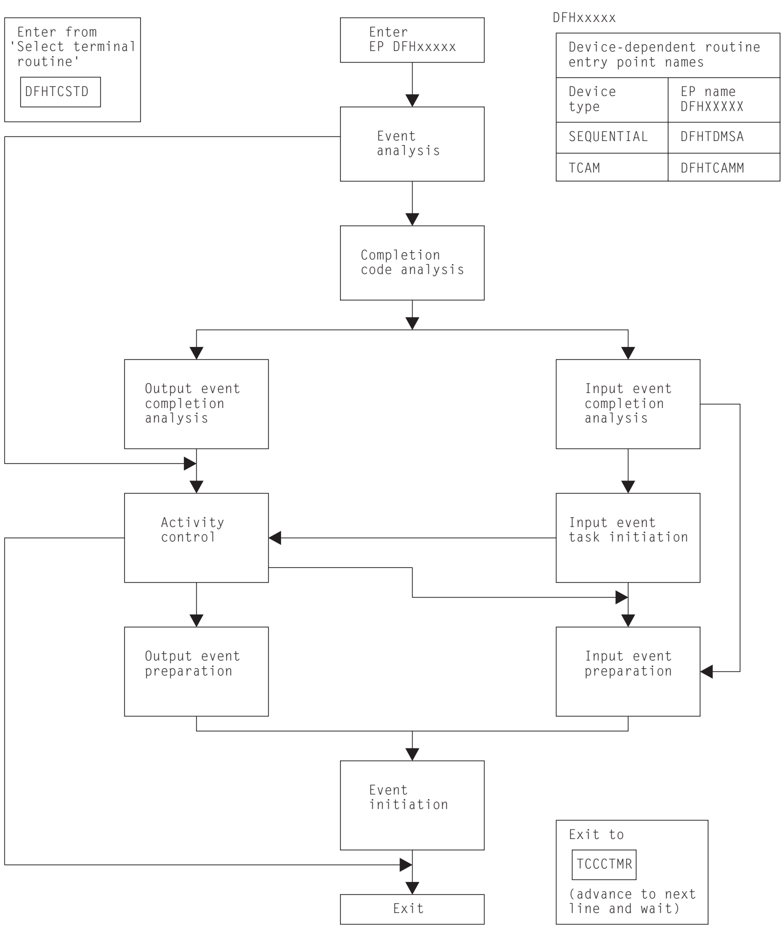


Figure 84. Terminal control general flow through device-dependent modules (TCP only)

High-performance option

When running CICS under MVS, the high-performance option (HPO) can be used. HPO uses VTAM with CICS as an authorized program so that the VTAM path length is reduced. This is achieved by dispatching SRBs to issue the send and receive requests for data to and from the terminals. The SRB code is executed in the DFHZHPRX module.

System console support

One or more MVS system consoles can be used as CICS terminals. This includes any MVS extended console introduced from MVS/ESA SP 4.1 onward; for example, a TSO user issuing the TSO CONSOLE command.

Each console has a unique number (releases prior to MVS/ESA SP 4.1) or a unique name (MVS/ESA SP 4.1 onwards). This matches the console number or name defined in the MVS system generation. Consoles are defined to CICS using CEDA DEFINE TERMINAL (see Chapter 42, "Resource definition online (RDO)," on page 343). The console number or name is specified using the CONSOLE or CONSNAME keyword respectively, depending on the level of MVS.

The console operator communicates with CICS using the MVS MODIFY command to start transactions. CICS communicates with the console using either the WTO macro or the WTOR macro.

A system console is modeled by CICS as a TCTTE that has an associated control block, the console control element (CCE). The CCE holds the event control block (ECB) for the console, and both the console ID and the console name.

The interface between a system console and CICS is the command input buffer (CIB), which is created in MVS-protected storage for each MODIFY command. A CIB contains the data for a MODIFY command. CICS addresses the first CIB using the EXTRACT macro and the CIBs are chained together.

The MVS communication ECB is in MVS-protected storage; it is posted complete for each MODIFY command and reset when there are no CIBs to be processed. The CICS system wait list holds pointers to the MVS communication ECB and the ECB for each system console.

When CICS is initialized, an EXTRACT macro is executed to obtain the job name and point to the MVS communication ECB and the first CIB; all these are stored in the TCT prefix.

DFHZCP contains two modules, DFHZCNA and DFHZCNR, which perform system console support.

DFHZCNA is used to:
- Resume a task on completion of a terminal event for the task
- Attach a task to satisfy a request for transaction initiation by a MODIFY command
- Attach a task (AVAIL) requested by automatic transaction initiation (ATI)
- Detach a terminal from a task when the task has completed
- Shut down console support when CICS is quiescing.

DFHZCNR is used to:
- Issue WTO macros for application program WRITE requests
- Issue WTO and WTOR macros for application program CONVERSE or (WRITE,READ) requests
- Issue a WTOR macro with message DFH4200 for application program READ requests.

Console support control modules

DFHZDSP calls DFHZCNA to scan the consoles for any activity.

DFHZCNA checks whether any task is suspended because it is waiting for a terminal event, for example, a READ, and, if the event is completed, resumes that task before starting any new task. This is done by scanning the CCE chain for ECBs that have been posted by MVS.

When a MODIFY command is executed, the communication ECB is posted complete and a CIB for the command is added to the end of the CIB chain. DFHZCNA processes the CIB chain in first-in, first-out order. For each CIB, DFHZCNA searches the CCE chain for the console. With MVS/ESA SP 4.1 (or later), the search is on console name; otherwise, the search is on console ID.

The task is then attached if the ‘task pending’ flag in the CCE is not set by a preceding CIB in the chain. In the course of scanning the CIB chain, DFHZCNA may find a MODIFY command that requires a task to be attached, but cannot attach the task immediately because there is already a task active, or there is an outstanding error condition to clear. DFHZCNA therefore sets the ‘task pending’ flag in the CCE to remember the existence of the CIB. During the CIB chain scan, the condition preventing the task attach might clear, and a subsequent CIB might be selected for attach. However, the ‘task pending’ flag prevents this, and ensures that CIBs are processed in order. All ‘task pending’ flags are reset before each CIB chain scan.

If the task is to be attached, DFHZCNA obtains a TIOA and moves the data from the CIB to the TIOA. DFHZATT is then called to attach the task. If the attach fails, the TIOA is freed. A QEDIT macro frees the CIB if the attach is successful, and the scan continues.

When a transaction is automatically initiated and DFHKCP schedules the transaction for a terminal which is a console, a flag is set in the CCE by DFHZLOC. After DFHZCNA has completed scanning the CIB chain, it checks that the console does not have a task already attached and there is not a CIB on the chain for the console; if both these conditions are satisfied, the task is attached.

DFHZCNA issues a QEDIT macro to prevent any more MODIFY commands being accepted when CICS is shutting down. Any MODIFY commands on the CIB chain after shutdown has been started are processed. When other access methods have been quiesced, and there are no tasks attached for a console, console support is shut down.

If a console not defined to CICS is used to enter a MODIFY command, DFHZCNA sets up an error code and links to DFHACP to issue the error message. This is done using the TCTTE for the error console, CERR.

DFHZCNR sends terminal control requests from an application program to a specific system console by issuing WTO and WTOR macros. It is called by DFHZARQ.

For a WRITE request, DFHZCNR executes either a single WTO macro, or one or more multiline WTO macros, depending on the amount of data specified for the request.

For a READ request, DFHZCNR acquires a TIOA for the reply area and executes a WTOR macro with a CICS-supplied message, DFH4200. This message requests the operator to reply, and the transaction waits for this reply.

For a CONVERSE or (WRITE,READ) request, DFHZCNR acquires a TIOA for the reply area and executes a WTOR macro with the data specified for the WRITE. If there is any data remaining, DFHZCNR then executes either a single WTO macro, or one or more multiline WTO macros, depending on the amount of data. The transaction then waits until the operator replies to this request.

Defining terminals to CICS

Terminal definitions are created as CSD records or DFHTCT macros (non-VTAM only) and then installed in (added to) the terminal control table (TCT) as TCT terminal entries (TCTTEs).

When a cold start is performed, CICS obtains its TCT entries from DFHTCT macros or from groups of resource definitions in the CSD file, which are named in the GRPLIST system initialization parameter. These are recorded in the CICS catalog.

When a warm start is performed, CICS obtains the definitions from the DFHTCT macros and from the CICS catalog; the GRPLIST is ignored.

On emergency restart, CICS obtains the definitions from the DFHTCT macros and from the CICS catalog; the GRPLIST is ignored. Then CICS re-applies any in-flight TCT updates using information from the system log.

During CICS execution, TCT entries can be added as follows:

- By using the CEDA INSTALL command
- By the autoinstall process when an unknown terminal logs on
- By the transaction routing component when a TCT entry is shipped from a terminal-owning to an application-owning region.
- By using the EXEC CICS CREATE command

During CICS execution, TCT entries can be deleted as follows:

- By using the EXEC CICS DISCARD command
- By the autoinstall process when an autoinstalled terminal logs off or has been logged for a period.

- By the transaction routing component when a TCT entry has been unused for a period.
- Using the CEDA INSTALL, EXEC CICS CREATE, transaction routing, or autoinstall processes to replace the old entry.

Figure 85 shows the terminal control table (TCT).

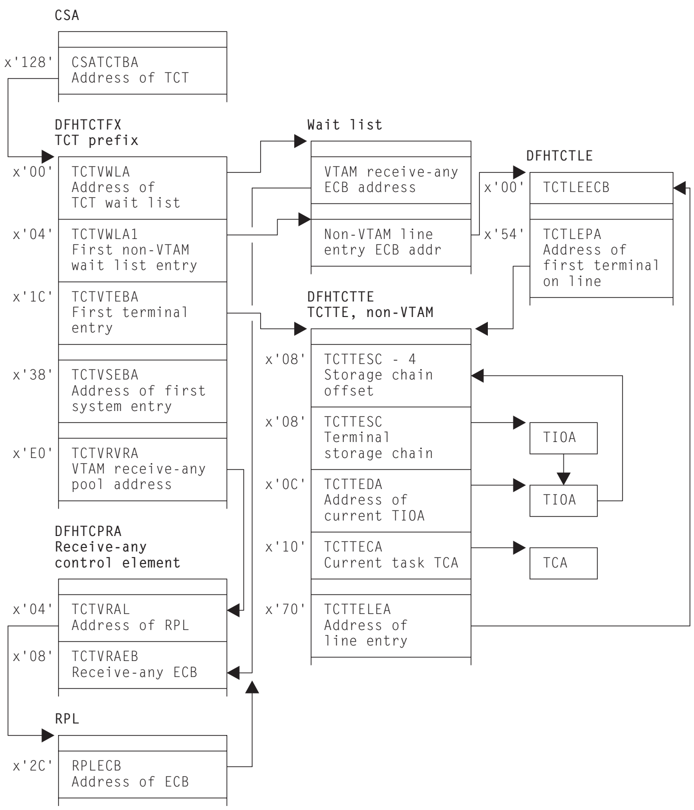


Figure 85. Terminal control table (TCT)

DFHZCQ

DFHZCQ installs, deletes, catalogs, uncatalogs, recovers, and inquires on terminals. Entries are installed in and deleted from the terminal control table by DFHZCQ. DFHZCQ is called by the following modules:

DFHAMTP
For the CEDA transaction and EXEC CICS CREATE, to install TCT entries

DFHEIQSC
For EXEC CICS DISCARD CONNECTION, to discard a connection.

DFHEIQST
For EXEC CICS DISCARD TERMINAL, to discard a terminal.

DFHTBSS
During CICS initialization, to restore terminal definitions at warm or emergency restart

DFHZATA
The autoinstall program

DFHZATD
The autoinstall delete program

DFHZATS
When a TCT entry is shipped, installed, or deleted for transaction routing

DFHZTSP
When a transaction route request is received to recatalog the connection if certain characteristics have changed.

DFHQRY
When the QUERY function is used to discover the actual characteristics of a device, complete the TCT entry, and recatalog the resulting TCTTE

DFHWKP
The warm keypoint program, to record information for RDO-eligible terminals in the CICS catalog, and to uncatalog autoinstalled entries.

DFHZCQ calls the table builder services (TBS) modules which in turn, call the appropriate DFHBSxxx modules to build the TCTTE for the input parameters. DFHZCQ is heavily dependent on the module that calls it to supply the complete set of parameters to be used to create the TCTTE; DFHZCQ itself is not responsible for determining parameters for the TCTTE.

DFHBS* builder programs

DFHZCQ calls the builder programs, whose names all begin DFHBS. These **builders** are responsible for creating TCTTEs. The parameters given to DFHZCQ are passed on to the builders, which extract the parameters and set the relevant fields in the TCTTE.

For further information about builders, see Chapter 6, “Builders,” on page 53.

Contents of the TCT

The TCT describes the logical units (LUs) known to CICS. Each active LU is represented by a terminal control table terminal entry (TCTTE). The TCT does not describe the network configuration; it describes the CICS logical viewpoint of the network.

The TCT contains pointers to these VTAM-related control blocks:

- Access method control block (ACB)— Link an application program, such as CICS, to VTAM
- Receive-any control blocks (RA-RPL, RA-ECB, RACE)— Process initial transaction input
- Node initialization block (NIB) descriptors and bind-area models— Used during logon processing
- TCTTEs— Describe the logical units known to CICS
- ACB and RPL exit lists— Point to the VTAM exit routines.

TCT indexing(DFHZGTI and DFHZLOC)

There are two types of requests that can be used in CICS to locate terminal entries:

1. DFHZGTI calls
2. and DFHTC CTYPE=LOCATE calls

Both these modules use DFHTM calls to a variety of indexes and chains to locate terminal entries in the TCT with efficiency.

The DFHZGTI module has the following call types:

Locate	Find a TCT entry in the given ‘domain’ which matches the name
GetStart	Obtain a browse token for Getnexts.
GetFirst	Find the first entry that matches the name in the given domain.
GetNext	Find the next entry that matches the name in the given domain.
GetEnd	Release the browse token
Release	Unlock an entry

Callers can decide to have an entry returned as locked or unlocked.

Terminal control

In DFHZGTI the total TCT is carved up into 'domains' A TCT entry can reside in several domains depending on its type. Callers to DFHZGTI specify one domain on a call and are returned one entry that fits the name (or partial name) that is supplied. DFHZGTI calls can be for the following domains:

Terminal by termid
All terminals (local, remote, non-vtam) by the terminal id (4-char).

Session by termid
All sessions (VTAM, MRO, remote) by the terminal id (4-char).

Global by termid
All terminal and all sessions by the terminal id (4-char).

System by sysid
All connections (local, remote) by the sysid (4-char)

MRO system by sysid
MRO connections by sysid (4-char).

LU61 system by sysid
LU61 connections by sysid(4-char).

REMDEL system by sysid
Systems that need REMDEL sent to them (because they do not support timeout) when a local entry is deleted by sysid (4-char).

Terminal by netname
VTAM local terminals by the netname (8-char).

System by netname
All connections (local, remote) by the netname (8-char).

Remote terminal by netname
Remote terminals by the netname (8-char).

Global by netname
Terminals, remote terminals and sessions by the netname (8-char).

Remote by Unique
All remote terminals and remote connections by the unique name that is Terminal-Owning-Region (TOR) netname, followed by a period, followed by the termid or sysid in the TOR. (13-char).

Remote terminal by Rsysid
Remote terminals by the value of REMOTESYSTEM (4-char).

Remote system by Rsysid
Remote connections by the value of REMOTESYSTEM (4-char).

Indirect system by Rsysid
Indirect connections by the value of REMOTESYSTEM (4-char).

Generic system by mbrname
Generic connections by the member-name of the connection in the generic VTAM resource (8-char).

DFHTC CTYPE=LOCATE calls are processed by DFHZLOC. DFHZLOC does not have access to as wide a range of domains as DFHZGTI, but it provides extra facilities such as finding particular types of sessions for a connection. Both DFHZGTI and DFHZLOC can lock TCT entries.

Locks

The table manager program (DFHTMP) is used to locate TCT entries by both DFHZGTI and DFHZLOC. When DFHTMP gives the address of an entry, it notes the address of the calling task, and this has the effect of a shared lock unless the caller asked for the entry not to be locked. All locks are released implicitly at the end of the task.

When a TCT entry is deleted, it must not be in use by another task. This is achieved by issuing the DFHTM QUIESCE macro. Other tasks that issue DFHTM LOCATE for that entry are suspended when they acquire a shared lock. These tasks are resumed when the original task issues a delete (if the commit option is used), or at syncpoint if not.

In addition to TMP read locks, DFHZLOC and DFHZGTI, use update locks which are obtained and released by DFHZGTA. DFHZGTA's involvement in TCT updates is discussed in Chapter 6, "Builders," on page 53. For efficiency, two flags in each TCT entry (one for delete and one for update) are examined before a TCT entry is returned. If either is set, and the request does not ask to see all updates, DFHZGTA is called to determine if the inquiring task holds the lock on the termid or sysid name. If it does, the entry is returned, otherwise the entry is ignored. This hides entries that are being installed or replaced from other parts of CICS until they are ready to be used, without requiring a lock search for each inquiry. The Builders, see Chapter 6, "Builders," on page 53, are responsible for setting and resetting the flags in the TCT entry.

The following sections describe some of the callers of DFHZCQ.

System initialization (DFHTCRP, DFHAPRDR and DFHTBSS)

The DFHTCRP program is responsible for reestablishing TCTTEs that were in existence in the previous CICS run. There are three stages of processing in DFHTCRP:

1. Initialize DFHZCQ and DFHAPRDR, then exit if START=COLD
2. Reestablish TCTTEs recorded in the CICS catalog calling DFHZCQ for each one.
3. Call DFHAPRDR to allow it to proceed and forward-recover in-flight updates to TCTTEs recorded in the system log at emergency restart or XRF takeover.

The DFHAPRDR program is called by DFHTCRP in two phases:
1. To initialize its control blocks.
2. To wait until Recovery Manager has delivered any inflight log records and DFHAPRDR (running on another task) has called DFHTBSS to recover them.

DFHAPRDR is called by Recovery Manager (RM) for each log record that are for UOWs that did not write a Forget record to the system log when CICS failed. It is then called again to denote the end of any such records. On this call DFHAPRDR waits until DFHTCRP has rebuilt the TCT from the catalog, and then calls DFHTBSS to recover each log record (which will update the TCT and catalog). Then it posts DFHTCRP to show that the TCT has recovered and returns to Recovery Manager.

The DFHTBSS program is called by DFHAPRDR with log records for TCT updates that were being written to the catalog when CICS failed. It then calls DFHZCQ to re-install or re-delete the entries that the log records represent.

CEDA INSTALL and EXEC CICS CREATE (DFHAMTP)

When the CEDA INSTALL command is used to install a group of TERMINAL definitions, the flow of control is as follows:
1. DFHAMP processes CEDA and EXEC CICS CREATE commands.
2. DFHAMPIL processes the INSTALL and CREATE commands.
3. DFHAMTP calls DFHTOR and then DFHZCQ.
4. DFHTOR receives as input a partial definition (TERMINAL, TYPETERM, CONNECTION, or SESSIONS), calling one of the DFHTOAxx modules, depending on the type of resource definition:
   - DFHTOAxx adds a partial definition to a BPS. For a terminal device, a complete BPS is built from information from one TYPETERM and one TERMINAL definition; for an ISC or MRO link, a complete BPSes are built from information from one CONNECTION and one (or more) SESSIONS definition(s).
   - DFHTOBPS builds the BPS, calling one of the DFHTRZxP modules to translate the parameter list into BPS format.
5. When DFHTOR has built a complete BPS, it returns it to DFHAMTP, ready to be passed to DFHZCQ.

For additional information about this process, see Chapter 42, “Resource definition online (RDO),” on page 343.

Autoinstall

For information about this process, see Chapter 3, “Autoinstall for terminals, consoles and APPC connections,” on page 11.

QUERY function (DFHQRY)

The QUERY function (DFHQRY) is used to determine the characteristics of IBM 3270 Information Display System devices, and complete the information about a device in the TCTTE. DFHQRY sends a read partition query structured field to the device, and analyzes the response. The TCTTE fields mainly affected are those used by basic mapping support (BMS), such as extended attributes. If QUERY(ALL) or QUERY(COLD) is specified in the terminal definition, DFHQRY is executed before any other transaction is initiated at a terminal. If QUERY(ALL) is specified, this is done after each logon. If QUERY(COLD) is specified, it is only done following the first logon after a cold start. After completing the TCTTE fields, DFHQRY calls DFHZCQ to recatalog the TCTTE.

Control blocks

Figure 86 shows the control blocks associated with terminal control.

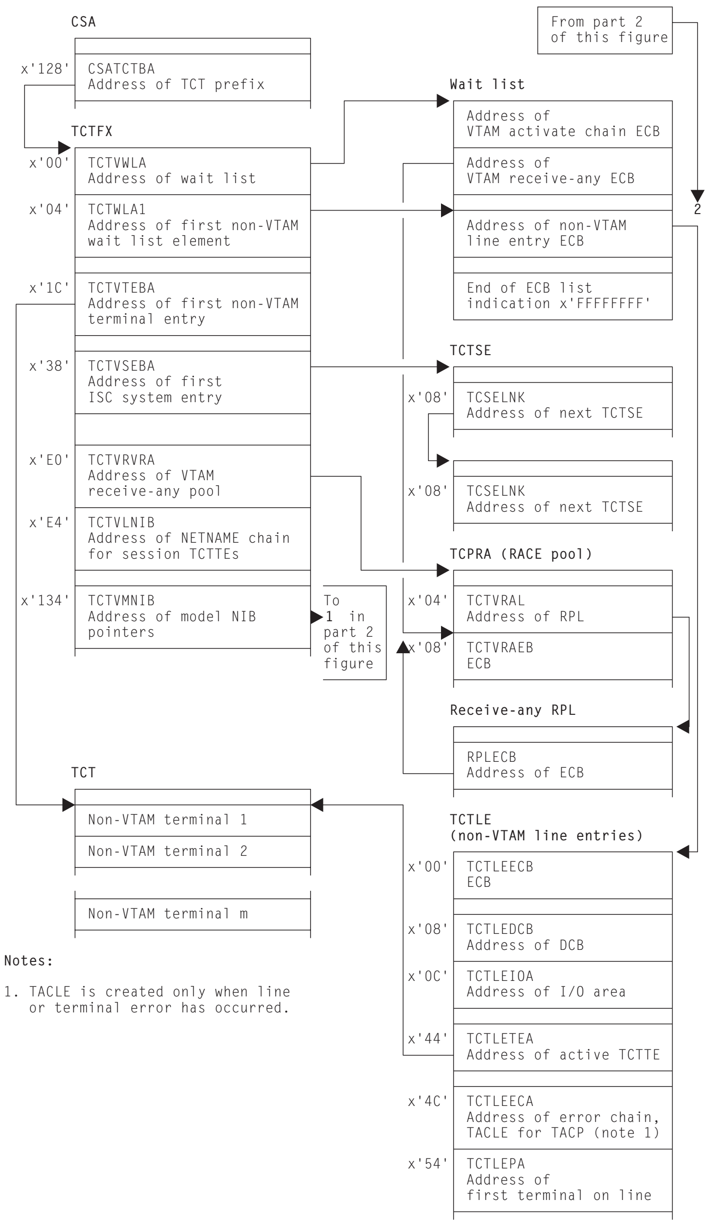
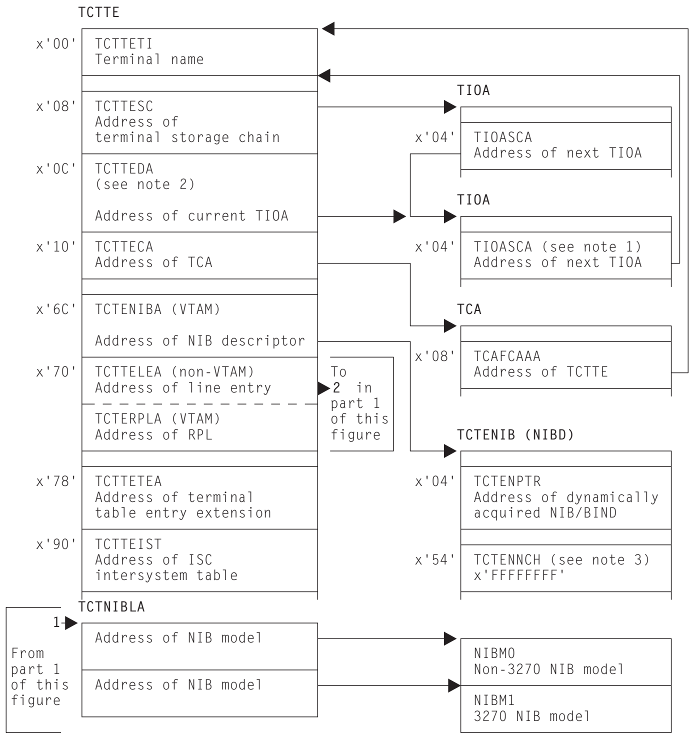


Figure 86. Control blocks associated with terminal control (Part 1 of 2)



Notes:
1. Chain field TIOASCA of the last TIOA in the chain addresses TCTTESC-4. The offset between TCTTESC-4 and TCTTESC is the same as the offset of TIOASCA in the TIOA.
2. TCTTEDA addresses the TIOA being used for the current I/O operation. This TIOA can be anywhere in the TIOA chain.
3. For session TCTTEs, TCTENNCH addresses the next NIBD on the NETNAME chain. Otherwise, TCTENNCH has the value x'FFFFFFFF', indicating that the NETNAME is in the TCNT (NETNAME table) managed by DFHTMP.

Figure 86. Control blocks associated with terminal control (Part 2 of 2)

Figure 87 shows the TCTLE and Figure 88 on page 422 shows the TACLE.

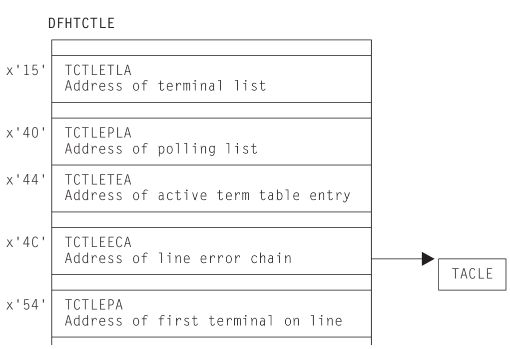


Figure 87. Terminal control table line entry (TCTLE)

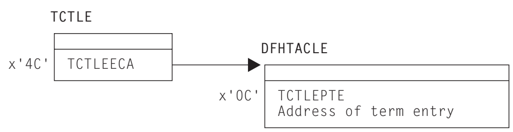


Figure 88. Terminal abnormal condition line entry (TACLE)

Terminal input/output areas (TIOAs) are set up by storage control and chained to the terminal control table terminal entry (TCTTE) as needed for terminal input/output operations. The TCTTE contains the address of the first terminal-type storage area obtained for a task (the beginning of the chain), and the address of the active TIOA.

See *CICS Data Areas* for a detailed description of these control blocks.

Modules

The DFHZCx modules contain CSECTs that issue VTAM macros to perform specific communication functions, and exit routines that are driven by VTAM when network events occur that are related to CICS.

The following is a list of the DFHZCx load modules concerned with terminal control and VTAM management in CICS, together with brief descriptions of their component object modules (CSECTs):

Module	CSECT	Description
DFHZCA	DFHZACT	Activate scan
	DFHZFRE	Freemain
	DFHZGET	Getmain
	DFHZQUE	Queue manager
	DFHZRST	RESETSR request
DFHZCB	DFHZATI	Automatic task initiation
	DFHZDET	Task detach
	DFHZHPSR	Authorized path SRB requests
	DFHZLRP	Logical record presentation
	DFHZRAC	Receive-any completion
	DFHZRAS	Receive-any slowdown processing
	DFHZRVS	Receive specific
	DFHZRVX	Receive specific exit
	DFHZSDR	Send response
	DFHZSDS	Send DFSYN
	DFHZSDX	Send synchronous data exit
	DFHZSSX	Send DFSYN command exit
	DFHZUIX	User input exit
DFHZCC	DFHZARER	Protocol error and exception handler
	DFHZARL	APPC application request logic
	DFHZARM	APPC migration logic
	DFHZARR	Application receive request logic
	DFHZARRA	Application receive buffer support
	DFHZARRC	Classify what next to receive
	DFHZARRF	Receive FMH7 and ER1
	DFHZBKT	Bracket state machine
	DFHZCHS	Chain state machine
	DFHZCNT	Contention state machine
	DFHZCRT	RPL_B state machine
	DFHZRLP	GDS post-VTAM receive logic
	DFHZRLX	GDS receive exit logic
	DFHZRVL	GDS pre-VTAM receive logic
	DFHZSDL	GDS send logic
	DFHZSLX	GDS send exit logic
	DFHZSTAP	Conversation state determination
	DFHZUSR	Conversation state machine

Module	CSECT	Description
DFHZCP	DFHZARQ	Application request handler
	DFHZATT	Attach routine
	DFHZCNA	MVS console
	DFHZDSP	Dispatcher
	DFHZISP	Allocate/free/point
	DFHZSUP	Startup task
	DFHZUCT	3270 uppercase translate
DFHZCW	DFHZERH	APPC ERP logic
	DFHZEV1	APPC bind security (part 1)
	DFHZEV2	APPC bind security (part 2)
DFHZCX	DFHSNAS	Create signon/sign-off ATI sessions
	DFHSNPU	Preset userid signon/sign-off
	DFHSNSU	Session userid signon/sign-off
	DFHSNTU	Terminal userid signon/sign-off
	DFHSNUS	US domain - local and remote signon
	DFHSNXR	XRF reflecting signon state
	DFHZABD	Abend routine for incorrect requests
	DFHZAND	Build TACB before issuing PC abends
	DFHZCNR	MVS console request
	DFHZIS1	ISC/IRC syncpoint
	DFHZIS2	IRC internal requests
	DFHZLOC	Locate TCTTE and ATI requests
	DFHZSTU	Status changing TCTTEs/LCDs and TCTSEs
DFHZCXR	DFHBSXGS	APPC session name generation
	DFHZTSP	Terminal sharing functions
	DFHZXRL	APPC command routing
	DFHZXRT	Routed APPC command handling
DFHZCY	DFHZASX	DFASY exit
	DFHZDST	SNA-ASCII translation
	DFHZLEX	LERAD exit
	DFHZLGX	LOGON exit
	DFHZLTX	LOSTERM exit
	DFHZNSP	Network services exit
	DFHZOPA	Open VTAM ACB
	DFHZRRX	Release request exit
	DFHZRSY1	Resynchronization part 1
	DFHZRSY2	Resynchronization part 2
	DFHZRSY3	Resynchronization part 3
	DFHZRSY4	Resynchronization part 4
	DFHZRSY5	Resynchronization part 5
	DFHZRSY6	Resynchronization part 6
	DFHZSAX	Send command exit
	DFHZSCX	SESSION control input exit
	DFHZSDA	Send command
	DFHZSES	SESSIONC
	DFHZSEX	SESSIONC exit
	DFHZSHU	Shutdown VTAM
	DFHZSIM	SIMLOGON
	DFHZSIX	SIMLOGON exit
	DFHZSKR	Send response to command
	DFHZSLS	SETLOGON start
	DFHZSYN	Handle CTYPE=syncpoint/recover request
	DFHZSYX	SYNAD exit
	DFHZTPX	TPEND exit
	DFHZTRA	Create ZCP/VIO trace requests
	DFHZXPS	APPC persistent session recovery
	DFHZXRC	XRF and persistent sessions state data analysis

Module	CSECT	Description
DFHZCZ	DFHZCLS	CLSDST
	DFHZCLX	CLSDST exit
	DFHZCRQ	CTYPE command request
	DFHZEMW	Error message writer
	DFHZOPN	OPNDST
	DFHZOPX	OPNDST exit
	DFHZRAQ	Read ahead queuing
	DFHZRAR	Read ahead retrieval
	DFHZTAX	Turnaround exit

Exits

DFHZCB has three global user exit points: XZCIN, XZCOUT, and XZCOUT1.

DFHZCP has one global user exit point: XZCATT.

DFHTCP has the following global user exit points: XTCIN, XTCOUT, XTCATT, XTCTIN, and XTCTOUT.

For further information about these, see the *CICS Customization Guide*.

Trace

The following point IDs are provided for terminal control:

- AP 00E6 (DFHTCP), for which the trace level is TC 2
- AP 00FC (DFHZCP), for which the trace level is TC 1
- AP FBxx, for which the trace levels are TC 1, TC 2 and Exc
- AP FCxx, for which the trace levels are TC 1, TC 2, and Exc
- AP FDxx, for which the trace level is TC 1
- AP FExx (APPC application receive requests), for which the trace levels are TC 2 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 57. Terminal error program

The terminal error program (DFHTEP) is invoked by the terminal abnormal condition program (DFHTACP) when an abnormal condition associated with a terminal or line occurs. The terminal error program (TEP) can be either of the following:

- The CICS-supplied sample TEP (DFHXTEP in source code form)
- A user-supplied TEP.

Design overview

The TEP analyzes the cause of the terminal or line error that has been detected by the terminal control program. The CICS-supplied version is designed to attempt basic and generalized recovery actions.

A user-supplied TEP can be used to enable processing to be performed whenever a communication system error is reported to CICS; for example, to analyze the error and accept or override the default actions set by DFHTACP.

When TEP processing is complete, control goes back to DFHTACP.

Note: Communication system errors (non-VTAM) are passed only to DFHTEP—not to the application programs.

Guidance information about TEP coding is given in the *CICS Recovery and Restart Guide*. Reference information about TEP coding is given in the *CICS Customization Guide*.

Modules

DFHTEP

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided specifically for this function; however, DFHTACP provides trace entries immediately before and after calling the terminal error program (see Chapter 55, “Terminal abnormal condition program,” on page 401 for further details).

Chapter 58. Trace control macro-compatibility interface

DFHTRP is responsible for handling all requests for trace services that are made by using the routine addressed by CSATRNAC in the CICS common system area (CSA).

Some parts of the CICS AP domain invoke DFHTRP to record trace information. This is achieved by use of the DFHTR, DFHTRACE, or DFHLFM macro.

DFHTRP converts all requests for recording trace entries into TRACE_PUT calls to the trace domain. All requests for changing the various trace flags that control tracing are converted into KEDD format calls to the kernel domain.

Design overview

The input to DFHTRP, set up by the macro used for the invocation or by the calling program directly, consists of the following TCA fields:

TCATRTR
: The trace request byte. The bottom half byte has one of the following values:
  - **2** User trace entry
  - **3** An entry requested via DFHLFM on entry to a LIFO module
  - **4** A system entry requested via DFHTR or DFHTRACE
  - **5** An entry requested via DFHLFM on exit from a LIFO module.

TCATRID
: The trace ID of the entry to be made. This is one byte X'nn'. The resulting trace point ID is AP 00nn.

TCATRF1/TCATRF2
: Two 4-byte fields to appear as FIELD A and FIELD B in the trace entry.

TCATRRSN
: An 8-character field used by some entries to specify a resource name.

The following flags in the TCA and CSA are tested by DFHTRP before making the call to the trace domain (TRACE_PUT function):

CSATRMAS (X'80' bit in CSATRMF1)
: The trace master flag. This is off unless at least one of internal, auxiliary, or GTF trace is active.

TCANOTRC (X'40' bit in TCAFLAGS)
: This is set according to the TRACE (YES|NO) specification on the TRANSACTION definition for the transaction ID used to start this task. It allows suppression of all trace activity for specified transaction IDs.

X'80' bit in TCATRMF
: This is the user entry 'single' flag. It allows suppression of user trace entries for the associated task.

The process flow is as follows:

1. Test appropriate flags and exit if trace not required.
2. Execute data collection routine specific to trace ID in TCATRID to set up fields in trace entry.
3. Call TR domain with TRACE_PUT call to write the entry to the active destinations.
4. Invoke the storage violation trap (if this has been activated) by using the CSFE DEBUG transaction, or by using the CHKSTSK or CHKSTRM startup override. See the *CICS Problem Determination Guide* for information about the detection of storage violations.

Modules

DFHTRP

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for trace entries recording “trace on” and “trace off” calls to DFHTRP:

- AP 00FE, for trace turned on
- AP 00FF, for trace turned off.

There are no corresponding trace levels for these point IDs; that is, the trace entries are always produced.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 59. Trace formatting

There are three possible destinations for CICS trace entries:

Internal
To main storage in the CICS region

Auxiliary
To a BSAM data set managed by CICS

GTF To the MVS-defined destination for generalized trace facility (GTF) records.

This section describes the code used to interpret and format CICS trace entries from all of these destinations when they are processed offline.

For more information about using traces in problem determination, see the *CICS Trace Entries*.

In this context, “formatting” is used to mean the overall process of producing a report, suitable for viewing or printing, from trace data in a dump or trace data set. “Interpretation” is the process of taking just the point ID and the data fields from a trace entry and producing a character string describing what the entry represents.

There are four environments for trace formatting:
- Internal trace in transaction dump
- Internal trace in system dump
- Printing auxiliary trace data set
- Printing GTF trace data set or processing GTF records in an SDUMP.

Table 27. CICS trace formatting summary

	Transaction dump printout	**System dump printout**	**Auxiliary trace printout**	**GTF trace printout**
CICS trace type	Internal	Internal	Auxiliary	GTF
Data set	DFHDMPx	SYS1.DUMPnn	DFHxUXT	SYS1.DUMPnn or SYS1.TRACE
Controlling program	DFHDU640	DFHTRDUF	DFHTRPRA	DFHTRPRG
Load module name	DFHDU640	DFHPD640	DFHTU640	AMDUSREF (alias DFHTR640)

Design overview

The controlling program (DFHDU640, DFHTRDUF, DFHTRPRA, or DFHTRPRG) is responsible for acquiring the trace formatting control area (TRFCA), which is used for communication between the different routines.

As far as possible, the necessary code is constructed of routines that can run in all four environments. Subroutines required by the common code that cannot themselves be common (such as the line print subroutine) have their addresses placed in the TRFCA by the controlling program.

The controlling routines are:

DFHDU640 The dump utility program used to print transaction dumps. Invokes DFHTRFPB for each internal table block.

DFHTRDUF The system dump formatting routine for the trace domain. Invokes DFHTRFPB for each internal table block.

DFHTRPRA
The main routine of the trace utility program DFHTU640 used to print an auxiliary trace data set. Invokes DFHTRFPP to encode selective print parameters. Invokes DFHTRFPB for each auxiliary trace block.

DFHTRPRG
The main routine of the GTF format appendage for CICS entries (format ID X'EF') AMDUSREF (alias DFHTR640). Invokes DFHTRFPP to encode selective print parameters. Invokes DFHTRFFE for each trace entry.

A noncommon subroutine required in all four environments is:

TRFPRL
Print a specified character buffer. This is contained in the controlling program.

The common routines required in more than one environment are:

DFHTRFPP
Process parameters. Passed a character string, encodes the string as selective print parameters into the TRFCA (for DFHTRPRA and DFHTRPRG only). See the *CICS Operations and Utilities Guide* for details of the selective print parameters.

DFHTRFPB
Process block. Processes a trace block from a dump or auxiliary trace data set, calling DFHTRFFE for each entry in the block.

DFHTRFFE
Format entry. Passed a trace entry, it calls DFHxxTRI, TRFPRL, and DFHTRFFD to produce the formatted entry.

DFHTRFFD
Format data. To format and print the trace data fields of a particular entry in hex and character form. Calls TRFPRL to print each line.

DFHxxTRI
The interpretation routine for the *xx* domain. Builds the interpretation string for a particular entry given the trace point ID and the data fields from the entry. The AP domain routine DFHAPTRI calls one of the interpretation routines DFHAPTRx. Each of these is responsible for a functional component of the AP domain.

DFHTRIB
The interpretation build program. Adds printable data to the interpretation buffer in the TRFCA as requested by the interpretation routine.

DFHCDCON
The interpretation of some trace entries requires analysis of domain call parameter lists. Converts a hexadecimal parameter list into a printable list of keywords. If the resulting interpretation string would have been more than 1024 bytes long if all keywords were included, the warning '<<INTERPRETATION OVERFLOWED>>' is printed with the string.

DFHxxyyT
The data file for an *xxyy* format parameter list that is used by DFHCDCON to translate the hexadecimal parameter list into a printable list of keywords.

The components of the trace formatting function are shown in Figure 89 on page 431.

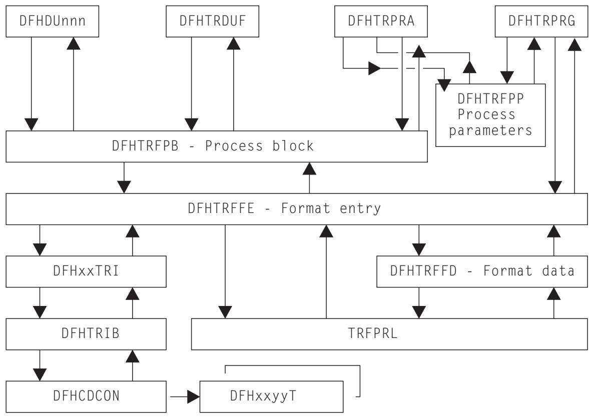


Figure 89. Trace formatting components

Segmented entries on GTF

GTF entries with the CICS format ID X'EF' are written from parts of CICS that run asynchronously with the mainline code, as well as from the trace domain itself. The source of the entry is identified by the type byte in TREN_TYPE in the entry header. See DFHTREN in the *CICS Data Areas* manual for a full description of the trace entry header.

```
Type      Source of entry
 00         TR domain
 01         not used
 02         DFHMNSVC
 03         'normal' CICS VTAM exit
 04         CICS VTAM LERAD/SYNAD exit
 05         CICS VTAM TPEND exit
 06         CICS VTAM HPO exit
 07         CICS VTAM HPO LERAD/SYNAD exit
```

For trace formatting, the different types run on different MVS threads. Because CICS entries can be split into several GTF entries due to the 256-byte restriction on GTF entry length, it is possible that header and continuation entries of the different types may be interleaved on the GTF data set. DFHTRPRG allows for this by having 4KB buffers for each type in which it can reconstruct segmented entries. This is made all the more relevant when it is recognized that there could be several CICS regions writing to the GTF data set at the same time. Not only may different types become interleaved, but also records of the same type but from different CICS regions. For each type there can be up to five 4KB buffers for reconstructing the segmented entries to ensure that all the entries for any region are formatted completely and correctly. This makes the segmenting of the entries transparent in a formatted GTF trace, although they appear in order of completion and so may be out of time sequence.

Control blocks

The trace formatting control area (TRFCA) is used as a communication area between the routines that go to make up each of the four trace formatting load modules. See the *CICS Data Areas* manual for details of DFHTRFCA.

Modules

Module	Function
Controlling programs	

Trace formatting

Module	Function
DFHDU640	Internal trace in transaction dump
DFHTRDUF	Internal trace in system dump
DFHTRPRA	Auxiliary trace
DFHTRPRG	GTF trace
Common routines	
DFHTRFPB	Process trace block
DFHTRFPP	Process selective print parameters
DFHTRFFE	Format trace entry
DFHTRFFD	Format data from entry
DFHTRIB	Interpretation build routine
DFHCDCON	Parameter list decode routine
Trace interpretation routines	
DFHAPTRA	MRO entries
DFHAPTRB	XRF entries
DFHAPTRC	User exit management entries
DFHAPTRD	DFHAPDM/DFHAPAP entries
DFHAPTRE	Data tables entries
DFHAPTRF	SAA communications and resource recovery entries
DFHAPTRG	ZC exception and VTAM exit entries
DFHAPTRI	Application domain entries (router)
DFHAPTRJ	ZC VTAM interface entries
DFHAPTRL	CICS OS/2 LU2 mirror entries
DFHAPTRN	Autoinstall terminal model manager entries
DFHAPTRO	LU6.2 application request logic entries
DFHAPTRP	Program control entries
DFHAPTRR	Partner resource manager entries
DFHAPTRS	DFHEISR trace entries
DFHAPTRV	DFHSRP trace entires
DFHAPTRW	Front End Programming Interface feature entries
DFHAPTR0	Old-style entries
DFHAPTR2	Statistics entries
DFHAPTR4	Transaction manager entries
DFHAPTR5	File control entries
DFHAPTR6	DBCTL entries
DFHAPTR7	Transaction routing entries
DFHAPTR8	Security entries
DFHAPTR9	Interval control entries
DFHCCTRI	Local and global catalog domain entries
DFHDDTRI	Directory manager entries
DFHDMTRI	Domain manager domain entries
DFHDSTRI	Dispatcher domain entries
DFHDUTRI	Dump domain entries
DFHKETRI	Kernel domain entries
DFHLDTRI	Loader domain entries
DFHLGTRI	Log Manager domain entries
DFHL2TRI	Log Manager domain entries
DFHLMTRI	Lock manager domain entries
DFHMETRI	Message domain entries
DFHMNTRI	Monitoring domain entries
DFHNQTRI	Enqueue domain entries
DFHPATRI	Parameter manager domain entries
DFHPGTRI	Program manager domain entries
DFHRMTRI	Recovery Manager domain entries
DFHSMTRI	Storage manager domain entries

Module	Function
DFHSNTRI	Signon entries
DFHSTTRI	Statistics domain entries
DFHTITRI	Timer domain entries
DFHTRTRI	Trace domain entries
DFHTSITR	Temporary Storage domain entries
DFHUSTRI	User domain entries
DFHXMTRI	Transaction manager domain entries
DFHXSTRI	Security domain entries

Exits

Global user exit points are not applicable to offline utilities.

Chapter 60. Transaction Failure program

The abnormal condition program has been divided into two new programs according to function.

1. **DFHTFP** which is a new program that is invoked after transaction initialization on abnormal termination.
2. **DFHACP** which is invoked by transaction manager whenever an incorrect transaction is detected.

The transaction failure program (DFHTFP) is invoked during transaction abend processing. Its purpose is to reset the status of a terminal attached to the transaction, and to send a message informing the terminal operator that the transaction has abended. It also calls the user-written (or default) program error program (DFHPEP), and writes a message to the CSMT transient data destination.

DFHTFP resolves any abnormal conditions other than those associated with a terminal, or those handled directly by the operating system.

Design overview

Errors can be classified as belonging in either of two broad categories:

1. **DFHTFP**. Task abnormal conditions, which are detected by CICS control programs and are often due to an application program destroying system control information. When this happens, the task is terminated, the program error program (DFHPEP) is called, the terminal operator is, if possible, informed of the error, and the error is logged at destination CSMT. If the transaction has entered syncpoint processing, then DFHPEP is **NOT** called.
2. **DFHACP**. Operator errors, such as incorrect transaction identifiers, security key violations, or failure of an operator to sign on to the system before attempting to communicate with CICS. When any of these happens, the program error program is **NOT** called, the terminal operator is notified, and the error is logged at destination CSMT.

Figure 90 on page 436 and Figure 91 on page 436 show the interfaces between the abnormal condition programs, DFHTFP and DFHACP, and other components when an error has been detected.

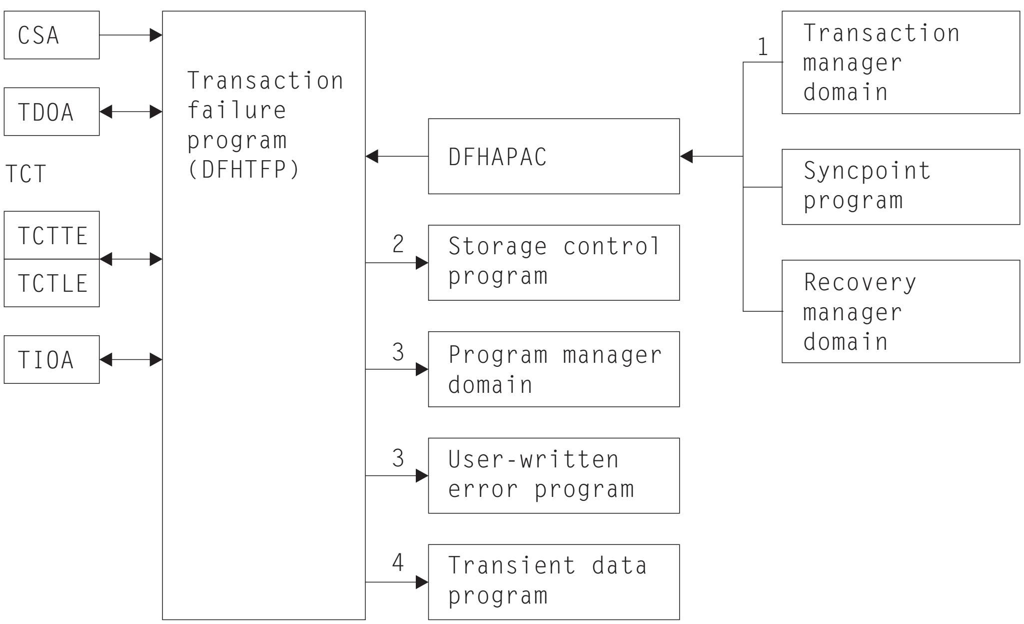


Figure 90. DFHTFP abnormal condition program interfaces

Notes:

1. DFHTFP is invoked by transaction manager whenever a task is abnormally terminated. The operator ID for error messages is in the terminal control table terminal entry (TCTTE) at TCTTEOI. DFHTFP returns to transaction manager after the error message has been issued. When a task is abnormally terminated because of a stall purge condition, the stall purge count is increased by one and the transaction identifier (from the installed resource definition) is included in the error message.
2. DFHTFP communicates with storage control to obtain and release terminal input/output areas (TIOAs).
3. DFHTFP links to the user-supplied (or default) program error program by issuing a DFHPGLU LINK_URM domain call, which passes a parameter list via a COMMAREA (mapped in this case by DFHPCOM TYPE=DSECT). Any abend within a DFHPEP program results in control returning to DFHTFP unless there is an active HANDLE ABEND for this program. See Chapter 39, “Program error program,” on page 337 for further information about the DFHPEP program.
4. DFHTFP and DFHACP both write error messages to the transient data destination, CSMT, by calling the message domain.

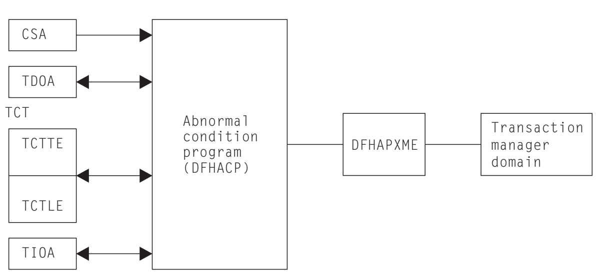


Figure 91. DFHACP abnormal condition program interfaces

Notes:

1. DFHACP is invoked by transaction manager whenever an incorrect transaction code is detected.
2. DFHTFP and DFHACP both write error messages to the transient data destination, CSMT, by calling the message domain.

Modules

DFHTFP, DFHACP, DFHAPAC, and DFHAPXME

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for the abnormal condition program:
- AP 00DC, for which the trace level is AP 1.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 61. Transaction restart program

The transaction restart program, DFHREST, is a user-replaceable program that helps you to determine whether or not a transaction is restarted. The default version of DFHREST requests a transaction restart under certain conditions; for example, if a program isolation deadlock occurs (that is, when two tasks each wait for the other to release a particular DL/I database segment), one of the tasks is backed out and automatically restarted, and the other is allowed to complete its update.

For further information about the transaction restart program, see *CICS Recovery and Restart Guide*. For information about how to provide your own code for DFHREST, see *CICS Customization Guide*.

Design overview

In the creation of the program control table (PCT), the system programmer can designate selected transactions as **restartable**.

During the execution of any transaction, certain temporary-storage data, intrapartition destinations, and files are protected for dynamic backout. In addition, for a restartable transaction, the following actions take place:

- Any terminal input/output area (TIOA), command-level communication area, or terminal user area existing at task initiation is copied to the dynamic log.
- Interval control automatic initiate descriptors (AIDs) used in the task are preserved by means of deferred work elements (DWEs) until the next syncpoint.
- Data is maintained to show:
  - What terminal traffic has occurred during the task
  - Whether a syncpoint has been passed
  - Whether or not the current activation of the task is the result of a restart.

If a transaction abends, but before backout has been attempted, DFHREST may be invoked to decide whether or not the task is to be restarted. Even if DFHREST decides that the transaction can be restarted, CICS may overrule the restart, for example because of a transaction backout failure.

DFHREST is invoked by DFHXMTA passing a parameter list via a COMMAREA that is mapped by the DFHXMRSD DSECT. DFHREST should return to DFHXMTA, indicating whether or not the transaction should be restarted. If DFHREST requests a restart, and CICS does not overrule this decision, the principal facility is not released and the principal facility owner reattaches a new task to restart the transaction.

Notes:

1. DFHREST can invoke CICS facilities such as file control and transient data, via the command-level interface.
2. If an error occurs while linking to, or in, the transaction restart program, the restart is not attempted for this task.
3. DFHREST runs before backout.

Control blocks

CICS supplies a description of the transaction restart program commarea, in Assembler-language, COBOL, PL/I, and C, which maps the layout of the parameter list passed between DFHXMTA and DFHREST. The parameter list contains information that helps you code your own version of DFHREST to determine whether a restart should be requested for a task.

For a detailed description of this control block, see the *CICS Data Areas*.

Modules

DFHREST is a skeleton user-replaceable program that you can modify.

Exits

Global user exit points are not relevant for this function.

Trace

Trace point IDs are not relevant for this function.

Statistics

CICS keeps a count of the number of times that each transaction has been restarted.

Chapter 62. Transaction routing

Transaction routing allows one CICS system to run a transaction in another CICS system. The transaction routing facility enables a terminal operator to enter a CICS transaction code into a terminal attached to one CICS system, and thereby start a transaction on another CICS system in a different address space in the same processing system or in another system.

There are two cases of transaction routing:

- Advanced program-to-program communications (APPC); that is, LU6.2
- Non-APPC (for example, LU2).

APPC transaction routing makes use of much of the non-APPC function, and there is often considerable overlap between the function provided by modules for each of the two cases.

The *CICS Intercommunication Guide* gives a detailed description of transaction routing.

Design overview

Figure 92 shows the overall design of this component.
CICS executes the CICS relay program DFHAPRT (which invokes the user-replaceable dynamic

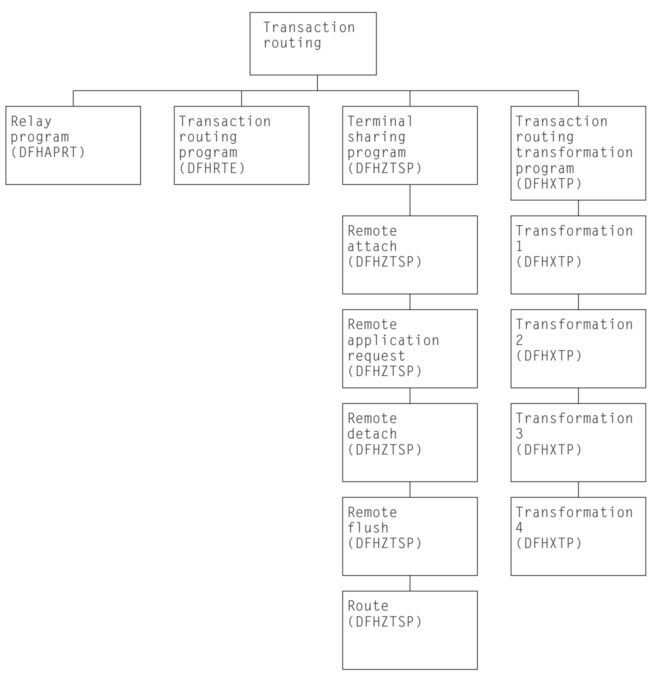


Figure 92. Transaction routing

transaction routing program) as follows:

- When a transaction defined with the value DYNAMIC(YES) is initiated.
- When a transaction definition is not found and CICS uses the special transaction defined on the DTRTRAN system initialization parameter. (For more information about DTRTRAN, see the *CICS System Definition Guide*.)
- Before routing a remote, terminal-oriented, transaction initiated by ATI.
- If an error occurs in route selection.
- At the end of a routed transaction, if the initial invocation requests re-invocation at termination.

If CICS has been generated with the appropriate options for intercommunication, the initialization of CICS with the ISC=YES system initialization parameter specified causes the following modules to be loaded:

- DFHXTP (transaction routing data transformation program)
- DFHZCXR (which includes the DFHZTSP CSECT, the terminal sharing program).

The entry point addresses of these modules are contained in the optional features list that is addressed by CSAOPFLA in the CSA.

The rest of this section is mainly concerned with APPC transaction routing, which occurs when an APPC device is linked through an LU6.2 session to a transaction that is defined as remote.

Overview of operation in the application-owning region for APPC transaction routing

Figure 93 shows the modules in the application-owning region for transaction routing for APPC devices.

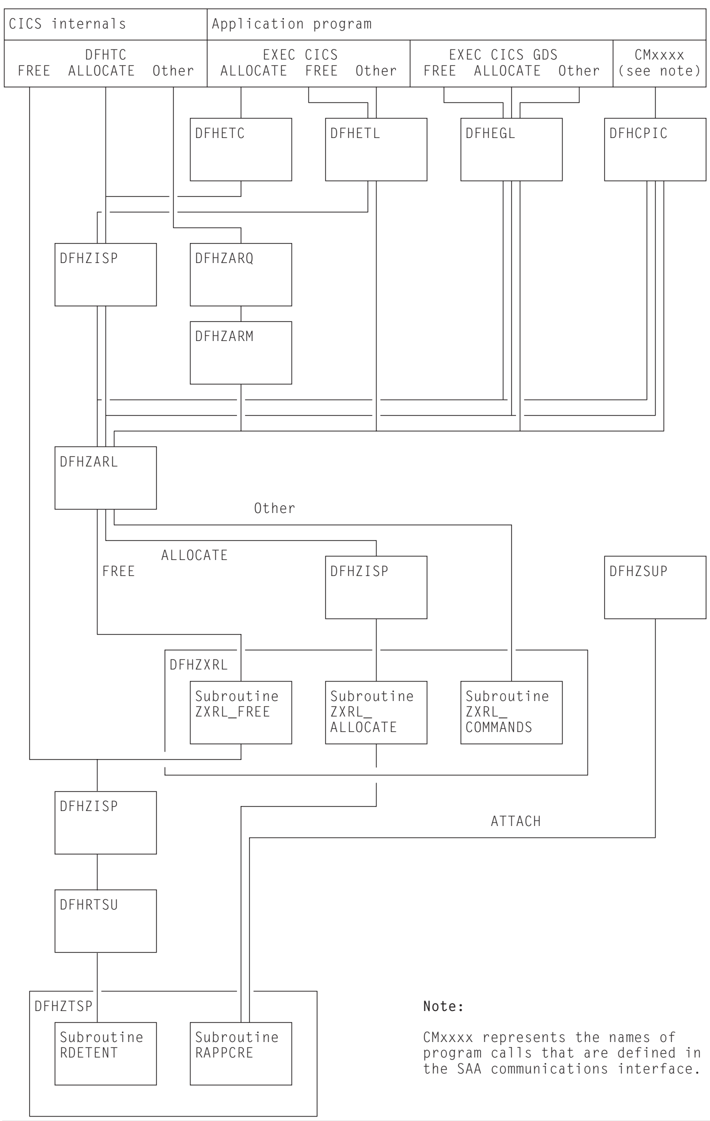


Figure 93. Transaction routing for APPC devices: modules in the application-owning region

APPC control blocks

A remote APPC device is defined in the application-owning region with a remote terminal control table system entry (or remote system entry). There are no TCT mode entries or session TCTTE entries associated with the remote system entry when it is defined.

A session with the remote APPC device is represented by a surrogate session TCTTE (or surrogate session entry). The surrogate is built dynamically when the conversation between the systems is initiated, and is deleted when the conversation terminates.

Figure 94 shows the way in which the TCT entries are related.

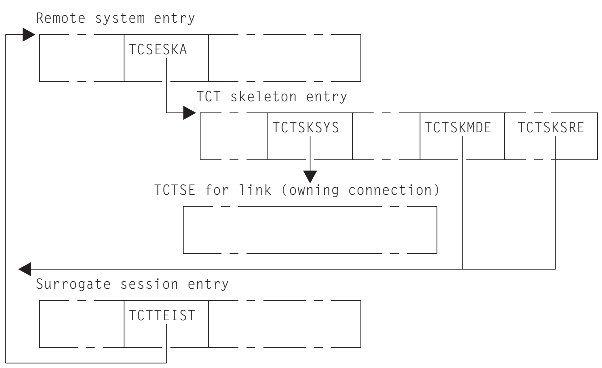


Figure 94. Transaction routing for APPC devices: TCT control-block structure in the application-owning region

Remote system entry: The remote system entry is similar to a normal system entry and, together with the TCT skeleton entry, also includes the following information:

- SYSIDNT of the terminal-owning region (TCTSKSYS)
- SYSIDNT of remote APPC device (local name) (TCTSKID)
- REMOTENAME of APPC device (SYSIDNT on terminal-owning region) (TCTSKHID)
- NETNAME of remote APPC device (TCSESID).

The remote system entry may be defined explicitly with CEDA DEFINE and INSTALL commands.

Alternatively, it is installed dynamically when the first transaction is routed from the remote APPC device. In this case, all data required to build the system entry is included in the initial ATTACH data stream from the application-owning region. No INQUIRE or INSTALL data is sent.

The remote system entry is recorded on the catalog and recovered after warm start and restart. It is located by TMP in the REMOTE domain and SYSTEM domain.

Surrogate session entry: The session between the terminal-owning region and the APPC device is represented in the application-owning region by a surrogate session entry.

The surrogate session entry is used to support the routing of commands to the APPC device, and to record security and status information for the conversation.

A surrogate session entry cannot be defined by the user; instead it is created when the conversation is initiated (by an ATTACH request from the APPC device, or an ALLOCATE request from the application-owning region), and is deleted when the conversation ends.

The surrogate session entry is not recorded on the catalog, is not accessible via TC LOCATE, and does not have an entry in the TMP index. It is not recovered after warm start or restart.

CEMT and EXEC CICS INQUIRE or SET commands cannot be used to modify a remote system entry.

DFHZXRL

This module forms a principal part of the transaction routing component for APPC devices. It passes DFHLUC macro requests issued in an application-owning region to the terminal-owning region.

All DFHLUC macro requests cause DFHZARL to be invoked. DFHZARL passes a request to DFHZXRL if the TCTTE address passed is for a surrogate session, and the request is one that DFHZXRL is known to handle (apart from ALLOCATE). ALLOCATE requests are always routed from DFHZARL to DFHZISP. DFHZISP is then responsible for calling DFHZXRL if the system from which a session is to be allocated is found to be remote. Table 28 summarizes this and shows which of the three main routines in DFHZXRL is called. ZXRL_ALLOCATE, ZXRL_COMMANDS, and ZXRL_FREE are described in "ALLOCATE processing in the application-owning region" on page 446, "Other LU6.2 command processing in the application-owning region" on page 447, and "FREE processing in the application-owning region" on page 447 respectively.

Table 28. DFHZXRL's processing of DFHLUC requests

DFHLUC request	DFHZXRL's caller	DFHZXRL routine called
ALLOCATE	DFHZISP	ZXRL_ALLOCATE
ISSUE-ABEND ISSUE-ATTACH ISSUE-CONFIRMATION ISSUE-ERROR ISSUE-SIGNAL RECEIVE SEND WAIT EXTRACT-PROCESS	DFHZARL	ZXRL_COMMANDS
FREE	DFHZARL	ZXRL_FREE

The input and output for DFHZXRL is provided by means of the LUC parameter list, that is, the parameter list which is built by the DFHLUC macro. DFHZARL passes the LUC parameter list to DFHZXRL unaltered. If the LUC parameter list previously contained only the SYSID name, DFHZISP adds the address of the remote system entry to the LUC parameter list before passing it to DFHZXRL.

DFHZXRL calls routine RAPPCRE of DFHZTSP to build the surrogate TCTTE representing the session with the APPC device, and DFHZISP calls routine RDETENT to free it.

ATTACH processing in the application-owning region

The following describes how a transaction is attached in the application-owning region when the attach request has been routed from the terminal-owning region.

DFHZSUP module:

1. Issues DFHSEC TYPE=CHECK,RESTYPE=TRAN to validate transaction security against the security values associated with the intersystem link at bind time.
2. Processes the incoming attach FMH5.

   For an LU6.2 ISC connection:

   - Sets the TCTTE to indicate a mapped or unmapped conversation.
   - Validates synclevel requested in FMH5 against the value negotiated at bind time.
   - Moves the TPN from the FMH5 to the TCA extension.
   - Performs attach-time security processing, as defined by the ATTACHSEC parameter in the resource definition for the LUC CONNECTION to the terminal-owning region. This may change the security

values associated with the link from the bind-time established values that were checked in step 1) to user-level values, obtained from the SNT for a userid specified in the FMH5.

For an MRO connection:

- Issues DFHZIRCT FN=ZSUP to extract the USERID and UOW-ID from the LU6.2 style FMH5.
- Performs attach-time security processing, as defined by the ATTACHSEC parameter in the resource definition for the LUC CONNECTION to the terminal-owning region. This can change the security values associated with the link from the bind-time established values that were checked in step 1) to user-level values, obtained from the SNT for a userid specified in the FMH5.
- Deletes the LU6.2-style FMH5 from the front of the data stream.

3. Issues DFHZUSRM TYPE=SET,REQUEST=ATTACH_INBOUND and DFHLUC TYPE=INIT-CALL macros to move input data into a buffer bypassing the FMH5 ATTACH header.
4. PIP processing is bypassed because PIP is never present on an attach from a terminal-owning region when transaction routing.
5. Puts the remaining data into a TIOA with a DFHTC TYPE=(READ,WAIT),NOATNI=YES.
6. Issues a DFHIS TYPE=RATT, to call DFHZTSP to build a surrogate session entry to represent the session TCTTE in the terminal-owning region.
7. Assign the security values established for the link to the surrogate, as preset security values are shipped from the terminal-owning region, and cannot be defined on the application-owning region.

   ATTACH security processing in DFHZSUP has established two SNTTEs associated with the link session:

   a. The SNTTE pointed to by TCTELSNT in the LU6.2 extension or TCTEIRSN for MRO represents link-level security values established at bind time.
   b. The SNTTE pointed to by TCTTESNT represents user-level security values established during ATTACH security processing.

   TCTTESNT is copied to the surrogate TCTTE. No provision is made for preset user security values to override the TCTTESNT value.

   Preset security values defined for the terminal session on the terminal-owning region are processed only on that system, during local attach processing. The SNTTE then associated with the local TCTTE is used to build the routed attach FMH5.

   At transaction end, no SNTTEs addressed by the surrogate are deleted when the surrogate is deleted. This is done, if necessary, as part of the termination of the LINK SESSION.

   Each system in a “daisy chain” imposes its own link security requirements. An intermediate system with a lower level of security would route the ATTACH with lower security (that is, no USERID or verified bit) which could cause it to be rejected by the next system in the chain.
8. Passes control to the requested application program.

DFHZTSP module:

1. Performs initialization housekeeping, checks the link TCTTE and TIOA.
2. Locates remote system entry from the TMP REMOTE domain. If not found, attaches the CITS transaction (DFHZATS) to install it.
3. Builds surrogate session TCTTE.
4. Gets a TIOA and chains it to the surrogate.
5. Issues DFHIS TYPE=XTP,XFNUM=2 to call DFHXTP.
6. Chains surrogate to TCA and Link TCTTE.
7. Copies link operator dispatching priority from the link and establishes dispatching priority for the surrogate.

DETACH processing in the application-owning region

At transaction end, routine RDETENT of DFHZTSP is called to delete the surrogate session entry. The remote system entry is not deleted, and can be used by a subsequent transaction routing request, by an ATI request, or by an ALLOCATE request issued in the application-owning region.

ALLOCATE processing in the application-owning region

A session can be allocated as a result of either of the following macro calls:

- DFHLUC TYPE=ALLOCATE
- DFHTC TYPE=ALLOCATE

The DFHLUC call invokes DFHZARL, which passes control to DFHZISP, the module that handles allocation and freeing of sessions. The DFHTC call invokes DFHZISP directly.

DFHZISP locates the TCTSE for the system identified on the ALLOCATE request.

The request is routed to DFHZXRL if the following conditions hold:

- The system is LU6.2
- The system is remote
- DFHZISP was called as a result of a DFHTC TYPE=ALLOCATE request (which is the case when DFHZISP is called from DFHZARL).

The address of the remote TCTSE is inserted in the parameter list passed to DFHZXRL.

If a Privileged Allocate request is made, the transaction abends, because the request is not permitted for a remote system.

DFHZXRL module: For an ALLOCATE request, control passes to subroutine ZXRL_ALLOCATE which establishes a session between the application-owning region and the alternate facility, and builds a surrogate session TCTTE.

Subroutine ZXRL_ALLOCATE:

1. Checks that the parameter list contains the TCTSE address for the remote LU6.2 system.
2. Obtains the address of the TCTSE of the system to which the LU6.2 commands are to be routed.
3. Allocates a session to the terminal-owning region.

   The connection between the terminal-owning region and application-owning region which supports remote alternate facilities may be an LU6.2 ISC connection or an MRO connection. Subroutine ZXRL_ALLOCATE allocates the session using a DFHTC TYPE=ALLOCATE macro call that can allocate a session on either type of connection.

   The default profile DFHCICSR is used; this may specify the modename for an LU6.2 connection. The modename specified on the EXEC CICS ALLOCATE is not used here, but is shipped to the terminal-owning region where it is used to allocate an LU6.2 session between the terminal-owning region and the APPC device.

   The queuing option (NOQUEUE|NOSUSPEND) specified on the ALLOCATE request by the caller is used when the DFHTC TYPE=ALLOCATE macro call is issued for the connection. If NOQUEUE is not specified, the request may also be queued when it is issued in the terminal-owning region. If a session failure occurs during this period, the transaction in the application-owning region and the relay transaction in the terminal-owning region abend.

   If a session between the application-owning region and terminal-owning region cannot be allocated:

   - When the failure is due to CICS logic, corruption of CICS storage, or incorrect resource definition by the user, the transaction abends.
   - When the failure is due to other conditions (such as session failure or ‘SYSBUSY’), an appropriate return code is passed to the caller.

The return code is handled so as to minimize the differences between local and remote APPC devices as seen by the user of the DFHLUC interface. The actions available are:
– Where the condition could be encountered with a local terminal, reflect the return code to the caller in LUCRCOD2 and LUCRCOD3 with LUCESYSI (X'01') in LUCRCOD1.
– Where the condition would not occur with a local terminal, reflect a different return code to the caller.

4. Issues a DFHIS TYPE=XTP,XFNUM=3 macro call that invokes a stream that is passed to the terminal-owning region.
5. Issues a DFHTC TYPE=(WRITE,WAIT,READ),FMH=YES macro call to send the request to the terminal-owning region and receive the response.
6. Issues a DFHIS TYPE=RALL that invokes DFHZTSP to build a surrogate session TCTTE, then chains the link session TCTTE and the surrogate session TCTTE together.
7. Issues a DFHIS TYPE=XTP,XFNUM=2 macro call that invokes DFHXTP to unwrap the response from the terminal-owning region and update the surrogate session TCTTE and the parameter list created by the DFHLUC macro.
8. Examines the return codes in the response:
   - If the request has been successful, returns the surrogate session TCTTE address to the caller.
   - If the request has not been successful, issues a DFHIS TYPE=RDET macro call to free the surrogate session TCTTE.

FREE processing in the application-owning region

One of the following macro calls is made in the application-owning region to request that a surrogate session TCTTE should be freed:
- DFHLUC TYPE=FREE
- DFHTC TYPE=FREE

The DFHLUC TYPE=FREE call invokes DFHZARL, which passes control to DFHZXRL; and subroutine ZXRL_FREE in DFHZXRL is then called to issue a DFHTC TYPE=FREE request against the surrogate. The DFHTC TYPE=FREE call invokes DFHZISP.

DFHZISP:
1. Bypasses security processing (sign-off) for a surrogate session entry, because the sign-off is performed for the link.
2. Issues the DFHIS TYPE=RDET macro that calls DFHZTSP to free the surrogate and link TCTTEs.

Other LU6.2 command processing in the application-owning region

Most SAA communications calls, EXEC CICS GDS commands, and EXEC CICS commands relating to LU6.2 sessions cause a call to DFHZARL using the DFHLUC macro.

The EXEC CICS SYNCPOINT, EXEC CICS SYNCPOINT ROLLBACK, and EXEC CICS (GDS) ISSUE PREPARE commands are handled under the control of the syncpoint program, which uses DFHLUC macro requests to send syncpoint flows on LU6.2 sessions, and DFHTC macro calls to end any dangling conversations.

DFHTC macro requests: DFHTC macro requests may be issued against surrogate session TCTTEs. Unlike requests for other surrogate TCTTEs, which are passed to DFHZTSP, DFHZARQ handles these requests in the same way as other requests against LU6.2 sessions: they are passed to DFHZARM which in turn calls DFHZARL. Within DFHZARL, requests are handled in a similar way to those initiated by the DFHLUC macro.

DFHLUC requests: DFHLUC requests are passed to DFHZARL: when the session is a surrogate, the request is passed to DFHZXRL (routine ZXRL_COMMANDS).

DFHZXRL module: Input to routine ZXRL_COMMANDS in DFHZXRL is the application command in the form of a DFHLUC macro call parameter list.

1. ZXRL_COMMANDS normally wraps up the command to be shipped and relevant TCTTE fields by calling a transformer routine in DFHXTP.

   However, if the first syncpoint flow has been received, then:

   - Application requests ISSUE-ERROR and ISSUE-ABEND are sent unwrapped on the link session.
   - All other requests are rejected with a state error.
2. ZXRL_COMMANDS tests the state of its link with the terminal-owning region (this may not be the same as the state of the application):

   If it finds that it is in ‘RECEIVE’ state, it issues a DFHTC TYPE=(READ,WAIT) in order to receive the change direction (CD) indicator from the terminal-owning region. Except during syncpoint processing, however, the session is normally in ‘SEND’ state when a command is issued.
3. ZXRL_COMMANDS then sends the wrapped-up request to the remote system using the DFHTC macro. To reduce the number of flows when the command may result in the termination of the conversation, the following rules are applied for both MRO and ISC links:

   - If the application command is SEND LAST WAIT and the application program is in ‘SEND’ state, the command is sent using a DFHTC TYPE=(WRITE,LAST) macro.
   - If the application command is WAIT and the application program is in ‘FREE PENDING AFTER SEND LAST’ state, the command is sent using a DFHTC TYPE=(WRITE,LAST) macro.
   - If the end bracket (EB) indicator has been sent to the terminal-owning region all other commands result in a state error return code.

   In other cases and when the link between the terminal-owning region and application-owning region is MRO, ZXRL_COMMANDS issues a DFHTC TYPE=(WRITE,WAIT,READ).

   However, when the link is LU6.2, the following additional rules are applied in order to exploit the buffering provided by LU6.2:

   - When the application’s command is a SEND and the application is in ‘SEND’ state ZXRL_COMMANDS, issues a DFHTC TYPE=(WRITE,WAIT) macro to send the request without waiting for a response.
   - When the application’s command is a SEND and the application is not in ‘SEND’ state ZXRL_COMMANDS, issues a DFHTC TYPE=(WRITE,WAIT,READ) so that it can get the state error back from the remote system immediately.
   - For all other commands, including SEND INVITE and so on, ZXRL_COMMANDS issues a DFHTC TYPE=(WRITE,WAIT,READ).
4. ZXRL_COMMANDS receives the response to its DFHTC macro call. This may be:

   - An ATNI or ATND abend. ZXRL_COMMANDS frees the link session and returns ‘TERMERR’ to the application.
   - ‘SIGNAL’, which is used by the terminal-owning region when it is in ‘RECEIVE’ state to indicate to the application-owning region that there is an abnormal response pending.

     ZXRL_COMMANDS issues a DFHTC TYPE=(WRITE,WAIT,READ) to send the change direction indicator and get the abnormal response from the terminal-owning region.
5. When the DFHTC macro included a READ, and the request was succesfully processed, ZXRL_COMMANDS checks for a wrapped reply from the terminal-owning region, and calls DFHXTP to unwrap the reply. When the resulting DFHLUC parameter list indicates SYNCPOINT or SYNCPOINT ROLLBACK, and the link is an MRO connection, ZXRL_COMMANDS issues a DFHTC TYPE=READ, because there is a SYNCPOINT or ROLLBACK flow pending.

   When there is no wrapped reply, ZXRL_COMMANDS checks for SYNCPOINT ROLLBACK received (the only possibility under these circumstances).

LU6.2 daisy-chaining considerations

There is no special-case code to distinguish between the terminal-owning region and an intermediate system. When DFHZXRT has interpreted a request received from the application-owning region, it issues the LU6.2 service request (DFHLUC) macro call with the parameter list that was created in the application-owning region. The macro generates a call to DFHZARL. If the TCTTE is a surrogate, which is the case in an intermediate system, control passes to DFHZXRL as described above.

Overview of operation in the terminal-owning region for APPC transaction routing

Figure 95 shows the modules in the terminal-owning region for transaction routing for APPC devices.

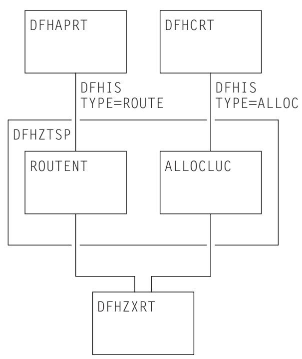


Figure 95. Transaction routing for APPC devices: Modules in the terminal-owning region

In the terminal-owning region, operation is under the control of a relay program. When transaction routing is initiated from the APPC device, the relay program is DFHAPRT (which is also used for non-APPC devices). When transaction routing is initiated by an ALLOCATE request in the application-owning region, the relay program is DFHCRT. Both relay programs call DFHZTSP, which calls DFHZXRT.

When an APPC device initiates a conversation with an application in the application-owning region, relay program DFHAPRT is started in the terminal-owning region. It calls the ROUTENT routine of DFHZTSP, which allocates a session to the application-owning region and starts the requested transaction there (see "ATTACH processing in the terminal-owning region").

When an application running in the application-owning region initiates a conversation with a remote APPC device by issuing an ALLOCATE request, the DFHCRT relay program is started in the terminal-owning region. It calls the ALLOCLUC routine of DFHZTSP which allocates a session to the APPC device (see Chapter 39, "Program error program," on page 337).

After a conversation has been started by either method, the LU6.2 commands passed from the application-owning region are processed by DFHZXRT, which issues the LU6.2 service request (DFHLUC) macro with an appropriate parameter list against the APPC device.

ATTACH processing in the terminal-owning region

The following flow describes the steps involved in routing a transaction from an APPC device across an LU6.2 intersystem link.

DFHZSUP module:

1. Processes the incoming FMH5 from the terminal. This:
   - Sets TCTTE to indicate mapped or unmapped conversation.
   - Validates synclevel requested in FMH5 against the value negotiated at bind time.
   - Moves the TPN from the FMH5 to the TCA extension.

- Performs attach-time security processing, as defined by the ATTACHSEC parameter in the resource definition for the APPC device (or CONNECTION). This may change the security values associated with the terminal from the default link-level values to user-level values, obtained from the SNT for a user who is signed on.

2. Checks transaction security code against new security levels developed during ATTACH security processing above.
3. Issues DFHSEC TYPE=CHECK,RESTYPE=TRAN to validate transaction security against the security values associated with the terminal (and with the user, if signed on).
4. Issues DFHZUSRM TYPE=SET,REQUEST=ATTACH_INBOUND and DFHLUC TYPE=INIT-CALL macros to move input data into a buffer bypassing the FMH5 ATTACH header.
5. If PIP is present, builds a new TCA extension and moves the PIP data into it by issuing a DFHLUC TYPE=RECEIVE (which also causes the PIP data to be deleted from the buffer).
6. Puts remaining mapped data into a TIOA with a DFHTC TYPE=(READ,WAIT),NOATNI=YES.
7. Issues DFHPC TYPE=XCTL to the relay program DFHAPRT.

DFHAPRT module:

1. Drives the dynamic routing exit if the transaction has been defined as dynamic.
2. Sets up the DFHISCRQ parameter list with remote sysid and tranid.
3. Recognizes that the principal facility is an APPC device.
4. Issues DFHIS macro to invoke DFHZTSP.

DFHZTSP module:

1. If the transaction has been defined with an associated TRPROF, the profile named is located with a DFHKC CTYPE=PROFLOC; otherwise the default DFHCICSS profile is used.
2. Issues DFHTC TYPE=ALLOCATE,REQUID=CSRR to allocate a session to the remote system using the profile identified in step 1.
3. Flags the returned TCTTE as a relay link and puts the remote sysid into TCTESYID in the terminal TCTTE. If the LINK TCTTE status is 'COLD', issues DFHTC CTYPE=CATALOG.
4. Sets up the transformer parameter list (DFHXTSTG) to indicate ATTACH FMH5 required, COLD or not COLD, and transaction routing for an APPC device, passing the tranid, user TCTTE, and link TCTTE.
5. Issues DFHIS TYPE=XTP,XFNUM=1 to call the transformer program, DFHXTP, to build the data. (See "Transformer program (DFHXTP)" on page 452.)
6. Issues DFHTC TYPE=(WRITE,WAIT,READ) against the link to route the ATTACH request to the application-owning region. This causes DFHZARM (when the link is ISC) or DFHZIS2 (when the link is MRO) to add an LU6.2 FMH5 preceding the LU6.1 FHM5 built by XTP. This contains security data required to validate the request at the application-owning region.

ALLOCATE processing in the terminal-owning region

DFHCRT module: Transaction CXRT (program DFHCRT) is started in the terminal-owning region when the attach FMH5 is received from the application-owning region

Program DFHCRT:

1. Checks that the principal facility of the task is an ISC or MRO session.

   If not, and if it is a terminal, a message is written to the facility, and the transaction terminates.
2. Issues DFHIS TYPE=ALLOC macro which calls DFHZTSP.

DFHZTSP module: The ALLOCLUC routine of DFHZTSP is invoked when the DFHIS TYPE=ALLOC macro is issued. This routine is called with input from the application-owning region in a TIOA.

Routine ALLOCLUC:

1. Issues DFHIS TYPE=XTP,XFNUM=4 which updates the TCTTE and builds a parameter list of the type created by the DFHLUC macro.
2. Verifies that the parameter list contains an ALLOCATE request (the only valid request at this stage). If it does not, the transaction abends.
3. Issues a DFHLUC MF=E macro with the supplied parameter list.
4. If the request is successful, DFHZTSP:
   a. Issues DFHIS TYPE=XTP,XFNUM=1 which wraps the updated TCTTE and DFHLUC parameter list ready for transmission to the application-owning region.
   b. Issues a DFHTC TYPE=(WRITE,WAIT,READ) against the session with the application-owning region.
   c. Passes control to DFHZXRT. The TIOA received with the preceding DFHTC request should contain data for one of the requests that DFHZXRT handles.
5. If the request is unsuccessful, DFHZTSP:
   - Issues DFHIS TYPE=XTP,XFNUM=1 which wraps the updated TCTTE and DFHLUC parameter list ready for transmission to the application-owning region.
   - Issues DFHTC TYPE=(WRITE,LAST) to send the response to the application-owning region.
   - Frees the session with the application-owning region.

FREE processing in the terminal-owning region

When an end-bracket has flowed from the application-owning region to the terminal-owning region as a result of an application command (for example, EXEC CICS SEND LAST), and the corresponding command has been issued in the terminal-owning region against the terminal, DFHZXRT issues a DFHLUC TYPE=FREE macro against the terminal, and a DFHTC TYPE=FREE macro against the link to the application-owning region.

Other LU6.2 command processing in the terminal-owning region

DFHZXRT is called by DFHZTSP following a DFHTC TYPE=(WRITE,WAIT,READ) macro. The reply received from the application-owning region is processed as follows:

1. If an application request has been received, DFHZXRT:
   - Calls DFHXTP to unwrap the application program's request
   - Issues the DFHLUC macro call with the parameter list created in the application-owning region
   - Calls DFHXTP to wrap the response to the DFHLUC macro
   - Sends the response to the application-owning region.

     Normally the wrapped terminal response is sent to the application-owning region with a DFHTC TYPE=(WRITE,WAIT,READ) macro. However, there are exceptions:
     - If the response to the DFHLUC macro call is a request for SYNCPOINT ROLLBACK, DFHZXRT sends the wrapped terminal response with a DFHTC TYPE=WRITE macro and then issues a DFHSP TYPE=ROLLBACK command.
     - If the response to the DFHLUC macro call is a request for SYNCPOINT, DFHZXRT sends the wrapped terminal response with a DFHTC TYPE=WRITE macro and then issues a DFHSP TYPE=PREPARE against the link.

       The response to the macro is processed in the same way as when a SYNCPOINT request is received from the application, and issued to the terminal, except that the roles of the terminal and link are reversed.
     - If the session to the terminal has been freed by an application command, DFHZXRT sends the wrapped terminal response with a DFHTC TYPE=(WRITE,LAST) macro.
     - When the session to the application-owning region is in 'RECEIVE' state, normally DFHZXRT issues a DFHTC TYPE=READ to get the next request from the application.

       However, if the link between the terminal-owning and application-owning regions is LU6.2, and the response to the DFHLUC macro issued to the terminal indicates that the terminal has issued one of ISSUE_SIGNAL, ISSUE_ERROR, ISSUE_ABEND, or SYNCPOINT_ROLLBACK, DFHZXRT issues an ISSUE_SIGNAL against the link with the application-owning region to notify the application-owning region that the terminal-owning region wants to send. It then issues a series of DFHTC TYPE=READ macros until it receives the change of direction indicator.

The data is processed in the normal way when 'SIGNAL' is received from the terminal. In the other cases, that is, if a negative response is received from the terminal, the data from the application-owning region is purged.

After the change direction indicator is received, DFHZXRT sends the response to the application-owning region, ISSUE_SIGNAL and ISSUE_ERROR are sent using a DFHTC TYPE=(WRITE,WAIT,READ) macro, ISSUE_ABEND is sent using a DFHTC TYPE=(WRITE,LAST) macro, and SYNCPOINT_ROLLBACK is sent using a DFHTC TYPE=WRITE macro.

– If the response from the terminal was 'ROLLBACK', by a DFHSP TYPE=ROLLBACK macro is issued.

2. If a syncpoint request has been received, DFHZXRT:
   - Issues a DFHLUC TYPE=ISSUE-PREPARE macro against the terminal TCTTE.
   - Checks the terminal's response:

     If the terminal response indicates that a SYNCPOINT or BACKOUT request was issued, DFHSPP is called.

     If the terminal response indicates that the terminal issued a SEND_ERROR request, DFHZXRT issues a DFHTC CTYPE=ISSUE_ERROR macro followed by a DFHTC TYPE=(WRITE,WAIT,READ) macro against the link session.

     If the terminal response indicates that the terminal issued DEALLOCATE(ABEND), DFHZXRT issues a DFHTC CTYPE=ISSUE_ABEND macro against the link session. It then frees the link with the application-owning region and returns.
3. If a syncpoint rollback request has been received, DFHZXRT issues a SYNCPOINT ROLLBACK request.

When DFHZXRT detects that EB has flowed on both the session with the terminal and the session with the application-owning region, it issues DFHTC TYPE=FREE on both and returns.

Transformer program (DFHXTP)

The terminal-sharing data-transformation program, DFHXTP, constructs and interprets the data streams flowing between terminal-owning and application-owning regions, for both APPC and non-APPC transaction routing environments.

It does this by using four transformers. These either wrap this data from the surrogate TCTTE (in the AOR) or the real TCTTE (in the TOR) into the link TCTTE's TIOA, or they unwrap this data from the link TCTTE's TIOA into the surrogate or real TCTTE.

The transformers work in matching wrap and unwrap pairs. Transformer 1 wraps any data to be sent from a TOR to an AOR, which is then unwrapped in the AOR by transformer 2. Transformer 3 wraps any data to be sent from an AOR to a TOR, which is then unwrapped in the TOR by transformer 4. Figure 96 on page 453 shows this process.

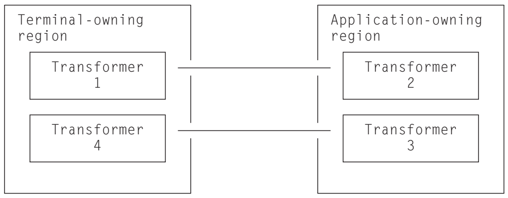


Figure 96. DFHXTP transformer operations

The transformer program is capable of shipping data from the TCTTE and the following control blocks that are chained off the TCTTE:

- The TCTTE extension, chained off TCTTETEA in the TCTTE.
- The terminal partition extension, chained off TCTTETPA in the TCTTE BMS extension.
- The TCTTE user extension, chained off TCTTECIA in the TCTTE.
- The SNTTE, chained off TCTTESNT in the TCTTE.
- The DFHLUC parameter list, and fields chained off it.

  Note that because this field is not chained off the TCTTE but is in LIFO, its address is passed as a parameter to the transformer program.
- The TCA extension for LU6.2 communication.
- Fields from the terminal control table system entry (TCTSE), chained off TCTTEIST in the TCTTE.
- Fields from the terminal control table mode entry (TCTME), chained off TCTTEMOD in the TCTTE.
- The data interchange block (DIB), chained off TCTEDIBA in the TCTTE.

The fields to be shipped are defined in tables in the transformer program.

There is special-case code to deal with fields that cannot be processed by the table-driven code.

For the transaction routing of LU6.2 commands, DFHXTP must ensure that the data stream built for transmission contains all the information relevant to support the issuing of a DFHLUC macro request on the remote system. This information consists primarily of:

- The DFHLUC parameter list
- Any data addressed by the parameter list
- The conversation state machine (TCTEUSRS in DFHTCTZE) in the TCTTE
- TCTTE fields required to build the surrogate TCTTE, in particular:
  - The synclevel supported by the terminal
  - The information returned to the application by the EXTRACT PROCESS command.

Data streams for transaction routing

Figure 97 on page 454 shows the types of transaction-routing data streams.

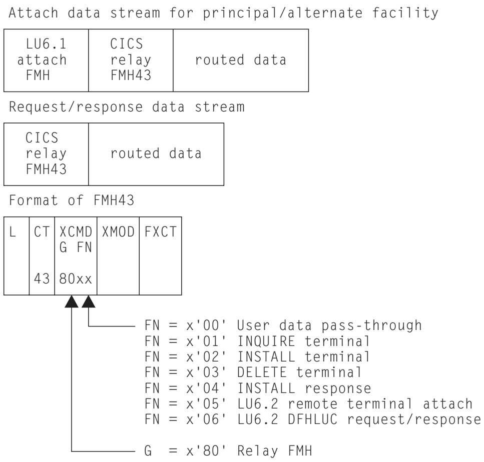


Figure 97. Transaction-routing data streams

The transformer builds four types of data stream for transaction routing:

1. Attach data stream for principal facility
   - Built by transformer 1
   - Shipped from TOR to AOR
   - Unwrapped by transformer 2
   - Contains an LU6.1 attach FMH (FMH5)
   - For LU6.2, the routed data does not contain a DFHLUC parameter list.
2. Attach data stream for alternate facility
   - Built by transformer 3
   - Shipped from AOR to TOR
   - Unwrapped by transformer 4
   - Contains an LU6.1 attach FMH (FMH5)
   - For LU6.2, the routed data contains a DFHLUC parameter list.
3. DFHLUC request data stream
   - Built by transformer 3
   - Shipped from AOR to TOR
   - Unwrapped by transformer 4
   - For LU6.2, the routed data contains a DFHLUC parameter list.
4. DFHLUC response data stream
   - Built by transformer 1
   - Shipped from TOR to AOR
   - Unwrapped by transformer 2
   - For LU6.2, the routed data contains a DFHLUC parameter list.

Note: The first transformer request for remote alternate facilities is to transformer 3, and not to transformer 1. This is because the same transformers are used whether transaction routing is initiated in the terminal-owning region or in the application-owning region.

An LU6.1 attach FMH5 is used when a transaction is to be started in the system to which the request is sent. CSRR is specified as the return process to indicate the use of transaction routing. In the case of routing to the application-owning region, the transaction is the user transaction; in the case of routing to the terminal-owning region, the transaction is the CXRT relay transaction.

Transaction-routed data format

Figure 98 shows the format of the data stream passed between a TOR and an AOR to provide transaction routing from any supported device.

The fields that are shipped depend principally on the type of terminal and on other parameters, as follows:

code	length	data	code	length	data	code	length	data	

Figure 98. Routed data format

The length field in Figure 98 depends upon whether the field type is described in the table that follows as being V (Variable), F (Fixed), or U (Undefined). A V field is 2 bytes in length, an F field is 1 byte, and U indicates a variable that is no longer wrapped or unwrapped if it is encountered.

Table 51 on page 753 shows the various data fields that may appear in a transaction routing data stream, together with their codes and field types.

Table 29. Transaction routing data stream. Built by the terminal sharing transformer (DFHXTP).

Code	Hex	Type	DSECT	Field	Description
1	01	V		XTPCDTC1	TC request bytes or attach start code
2	02	V		XTPCDOPC	Operator class
3	03	V		XTPCDTUA	TCTTE user area
4	04	V		XTPCDTIA	Terminal I/O area
5	05	V		XTPCDCMA	COMMAREA
6	06	V		XTPCDLPS	Terminal partition set
7	07	V		XTPCDPLM	Page LDC mnemonic
8	08	V		XTPCDPGD	Page data
9	09	V		XTPCDRQI	Request ID
10	0A	V		XTPCDETI	Error terminal ID
11	0B	V		XTPCDETL	Error terminal LDC
12	0C	V		XTPCDMCF	Message control flags
13	0D	V		XTPCDTTL	Message title
14	0E	V		XTPCDRTT	Route target ID: netname.termid.ldc.opid
15	0F	V		XTPCDCPS	Application partition set
16	10	F	DFHTCTTE	TCTTEAID	Automatic initiate descriptor
17	11	F	DFHTCTTE	TCTTECAD	Cursor address
18	12	F	DFHTCTTE	TCTESIDO	Outbound signal data
19	13	F	DFHTCTTE	TCTESIDI	Inbound signal data
20	14	F	DFHTCTTE	TCTE32SF	Screen size attributes
21	15	F	DFHTCTTE	TCTTEFX	Transparency attributes
22	16	F	DFHTCTTE	TCTTEBMN	Map set name
23	17	F	DFHTCTTE	TCTTECRE	Request completion extension
24	18	F	DFHTCTTE	TCTTECR	Request completion analysis
25	19	F	DFHTCTTE	TCTTEDES	TCAM destination name
26	1A	F	DFHTCTTE	TCTTETM	Terminal model number
27	1B	F	DFHTCTTE	TCTTETID	Teller identification for 2980
28	1C	F	DFHTCTTE	TCTTEOI	Operator identification
29	1D	F	DFHTCTTE	TCTTEEDF	EDF mode
30	1E	F	DFHTCTTE	TCTTETC	Nominated transaction
31	1F	F	DFHTCTTE	TCTTETS	Terminal status
32	20	U	DFHSNTTE	SNTESSF	Userid
33	21	F	DFHTCTTE	TCTEASCZ TCTEASCL TCTEASCC	Alternate screen size attributes
34	22	F	DFHTCTTE	TCTE32EF TCTE32E2	3270 extended feature flags
35	23	F	DFHTCTTE	TCTETXTF	3270 text feature flag

Table 29. Transaction routing data stream (continued). Built by the terminal sharing transformer (DFHXTP).

Code	Hex	Type	DSECT	Field	Description
36	24	F	TCTTETTE	TCTEAPGL TCTEAPGC	Alternate page size
37	25	F	DFHTCTTE	TCTECSG1 TCTECSG2	Coded graphic character set identifiers
38	26	F	DFHTCTTE	TCTEUSRS	LU6.2 conversation state machine
39	27	F	TCTTELUC	TCTECVT	LU6.2 conversation type (mapped or unmapped)
40	28	F	TCTTELUC	TCTESPL	LU6.2 syncpoint level
41	29	F	DFHTCTTE	TCTESPSA	Additional syncpoint flags
42	2A	F	TCTTELUC	TCTEIAHB	Attach FMH indicator
43	2B	F	DFHTCTSE	TCSESID	NETNAME of APPC device
44	2C	U	DFHSNTTE	SNTENLS	User's national language
45	2D	F	DFHTCTTE	TCTENLS	National Language Support Code
46	2E	F	DFHTCTTE	TCTESCFL	Security flag
47	2F	F	DFHTCTTE	TCTEITRS	Trace flags
48	30	F	DFHTCTME	TCMEMODE	Mode group name
49	31	F	DFHTCTTE	TCTTENLI	National language in use
50	32	F	TCTTELUC	TCTELUC1	LUC flag byte 1
51	33	F	DFHTCTTE	TCTESSPL	Synclevel of link
53	35	F	DFHTCTTE	TCTEVTP	Send mode/receive mode
54	36	F	DFHTCTTE	TCTTEIO	Task to be initiated
55	37	F	DFHLFS	PRESETC	Preset userid
56	38	F	TCTTETTE	TCTTEFMB	Outbound formatting status
57	39	F	DFHTCTTE	TCTEUCTB	UCTRAN = YES
58	3A	F	DFHTCTTE	TCTETSU3	UCTRAN = TRANID
63	3F	F	DFHTCTTE	TCTTETT	Terminal type code
64	40	F	DFHLUCDS	LUCOPN0 LUCOPN1 LUCOPN2 LUCOPN3	LUC request codes
65	41	F	DFHLUCDS	LUCRCODE	LUC request error feedback
66	42	F	DFHLUCDS	LUCSDBLK	LUC conversation feedback
67	43	F	DFHLUCDS	LUCNSYS	System name for LUC Allocate
68	44	F	DFHLUCDS	LUCMODNM	Modename for LUC Allocate
69	45	F	DFHLUCDS	LUCMSGNO	Message number for LUC Abend and Error
70	46	F	DFHLUCDS	LUCSENSE	Sense code for LUC Abend and Error
71	47	F	DFHLUCDS	LUCRQCON	Conversation type for LUC Issue Attach
72	48	F	DFHLUCDS	LUCRQSYN	Syncpoint level for LUC Issue Attach
73	49	F	DFHLUCDS	LUCFTPNL LUCFTPN	TPN for LUC Issue Attach
74	4A	F	DFHLUCDS	LUCPIP	PIP indicator for LUC Issue Attach
75	4B	F	DFHLUCDS	LUCTAREL	Maximum receivable length for LUC Receive
76	4C	F	DFHLUCDS	LUCMGAL	Mode group name of allocated session
90	5A	F	DFHDIBDS	DIBSENSE	DIB system/user sense data
128	80	V		XTPCDZIR	ZC install response
129	81	V		XTPCDZBP	ZC builder parameter set
130	82	V		XTPCDZIM	ZC install message set
131	83	V		XTPCOPCL	Opclass in routed message
132	84	V		XTPCDPNM	Program name for ISSUE LOAD
133	85	V		XTPLUCSD	Message text for LUC Send
134	86	V		XTPLUCRD	Message text for LUC Receive

Table 29. Transaction routing data stream (continued). Built by the terminal sharing transformer (DFHXTP).

Code	Hex	Type	DSECT	Field	Description
135	87	V		XTPLUTCX	TCA extension for LU6.2
136	88	V		XTPLUMSG	Message text for LUC Issue Abend or Issue Error
137	89	V		XTPIPASS	Issue Pass
138	8A	V		XTPLDATA	Logon Data
139	8B	V		XTPRETC	Issue Pass Return Code
140	8C	V		XTPLMOD	Issue Pass Logmode

Control blocks

Relay transaction control blocks

To support transaction routing, the relay transaction owns two TCTTEs; see Figure 99. One TCTTE is for the terminal, the other is for the link to the user transaction. The link TCTTE has bit TCTERLT in field TCTETSU set on, to indicate that it is being used by the relay transaction.

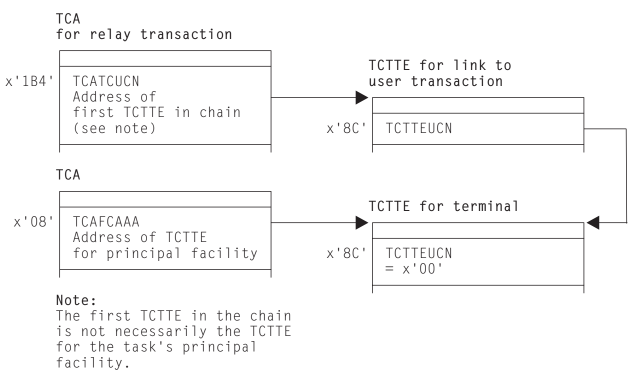
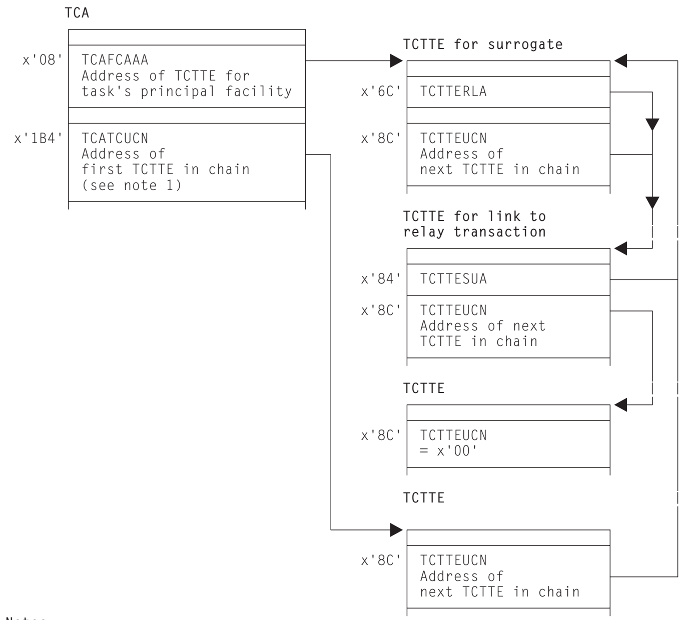


Figure 99. Control blocks associated with the relay transaction

User transaction control blocks

The user transaction owns two or more TCTTEs; see Figure 100 on page 458. One TCTTE is always present for the link to the relay transaction, and another TCTTE, called the surrogate TCTTE, represents the terminal TCTTE in the relay transaction address space. Field TCTTERLA in the surrogate TCTTE contains the address of the TCTTE for the link to the relay transaction. Bit TCTESUR (in field TCTETSU) set on indicates that the TCTTE is for a surrogate terminal. The link TCTTE has bit TCTERLX in field TCTETSU set on, to indicate that it is being used as a relay link.

If the user transaction executes CICS functions that are shipped to another address space or processing system, one TCTTE is chained off from the TCA for each different address space or processing system.



Notes:
1. The first TCTTE in the chain is not necessarily the TCTTE for the task's principal facility.
2. Apart from the surrogate and the link to the relay transaction, other TCTTEs can be in use for function shipping or DTP.

Figure 100. Control blocks for the user transaction (non-APPC device)

See the *CICS Data Areas* manual for a detailed description of these control blocks.

Modules

The principal modules associated with transaction routing are as follows:

DFHAPRT
is the relay program for non-APPC devices, and for APPC devices when the device initiates a transaction by sending an attach FMH5 to CICS.

DFHCRT
is the relay program for APPC devices when CICS sends an attach FMH5 to the device.

DFHRTSU
is the program which maintains the state of a surrogate APPC session during syncpoint

DFHXTP
is the data transformation program for terminal sharing. It constructs and interprets data streams flowing between terminal-owning and application-owning regions, for both APPC and non-APPC transaction routing environments.

DFHZTSP
is the terminal sharing program. It is used by transaction routing for devices of all types, exclusively so for non-APPC devices.

DFHZXRL
runs in the application-owning region to route APPC requests to the terminal-owning region.

DFHZXRT
runs in the terminal-owning region to receive APPC requests from the application-owning region, and issue them to the APPC device.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:
- AP DBxx (DFHXTP), for which the trace level is IS 1
- AP 08xx (DFHCRT, DFHZXRL, and DFHZXRT), for which the trace levels are IS 1, IS 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 63. Transient data control

Transient data control provides an optional queuing facility for managing data being transmitted between user-defined destinations (I/O devices or CICS tasks). This function facilitates data collection.

Design overview

The transient data program provides a generalized queuing facility enabling data to be queued (stored) for subsequent internal or offline processing. Selected units of information can be routed to or from predefined symbolic queues. The queues are classified as either **intrapartition** or **extrapartition**.

Intrapartition queues

Intrapartition queues are queues of data, held in a direct-access data set, for eventual input to one or more CICS transactions. Intrapartition queues are accessible only by CICS transactions within the CICS address space. Data directed to or from these internal queues is called intrapartition data. It can consist of variable-length records only.

An intrapartition queue is mapped onto one or more control intervals in the intrapartition data set. The control intervals are allocated to a queue as records are written and freed automatically as they are read or as the queue is deleted.

Examples of the data queued for intrapartition processing are:

- Transactions that require processes to be performed serially, not concurrently. An example of this type of process is one in which pending order numbers are to be assigned.
- Data to be used in a data set (file) update that could pass through the queue to allow the data to be applied in sequence.

Recovery of intrapartition transient data queues

Following abnormal system termination, intrapartition queues defined as recoverable by the user can be restored. Recovery is accomplished by reconstructing the queues from catalog data and from log records written automatically by CICS during normal execution. Two types of recovery are possible: **physical** and **logical**.

Physical recovery of intrapartition transient data queues: Physically recoverable transient data queues are restored to the state they were in when the system terminated abnormally. A physically recoverable transient data queue is not backed out if it has been updated by a unit of work (UOW) that has subsequently failed. Data written to such a queue is always committed and is restored during warm and emergency restarts.

When a UOW reads, writes, or deletes a physically recoverable queue, a log record is written to the system log. When the system is brought up after an abnormal termination, CICS can recreate a queue by retrieving definition information associated with the queue from the catalog, and state data from the log. .

Note: There is an exception to the rule that states that a physically recoverable queue is restored to the state it was in when CICS abnormally terminated. If a UOW reads a physically recoverable queue and CICS then terminates abnormally, the read operation will be backed out when CICS is subsequently brought back up.

Logical recovery of intrapartition transient data queues: Logically recoverable transient data queues are restored to the state they were in at the time they were last syncpointed. All inflight UOWs are backed out. If a UOW updates a logically recoverable queue and subsequently fails, all updates to the queue are backed out. Logically recoverable queues are restored during warm and emergency restarts.

Logically recoverable queues are logged as part of the first phase of syncpoint processing. When CICS is brought up after an abnormal termination, it can recreate logically recoverable queues by retrieving definition information associated with the queue from the catalog, and state data from the log.

Logically recoverable transient data queues can suffer from indoubt failures. If a UOW is indoubt and CICS abnormally terminates, the indoubt UOW environment is recreated when CICS is next brought up. When the indoubt failure is resolved, the UOW is committed or backed out.

Extrapartition queues

Extrapartition queues are sequential data sets on tape or direct-access devices. Data directed to or from these external queues is called extrapartition data and can consist of sequential records that are fixed- or variable-length, blocked or unblocked.

Data can be placed on an extrapartition data set by CICS for subsequent input to CICS or for offline processing. Sequentially organized data created by other than CICS programs can be entered into CICS as an extrapartition data set. Examples of data that might be placed on extrapartition data sets are:
- System statistics
- Transaction error messages
- Customer data, such as cash payments that can be applied offline.

Indirect queues

Intrapartition and extrapartition queues can be referenced through indirect destinations. This provides flexibility in program maintenance. Queue definitions can be changed, using the CEDA transaction, without having to recompile existing programs.

Automatic transaction initiation

When data is sent to an intrapartition queue and the number of entries (WRITEQs from one or more programs) in the queue reaches a predefined level (trigger level), the user can optionally specify that a transaction be automatically initiated to process the data in that queue.

The automatic transaction initiation (ATI) facility allows a user transaction to be initiated either immediately, or, if a terminal is required, when that terminal has no task associated with it. The terminal processing status must be such that messages can be sent to it automatically. Through the trigger level and automatic transaction initiation facility, an application program can switch messages to terminals. After a task has been initiated, a command in the application program is executed to retrieve the queued data. All data in the queue is retrieved sequentially for the application program.

Trigger transactions may only execute sequentially against their associated queue. When a trigger transaction has been attached, another transaction will not be attached until the first transaction has completed. If a trigger transaction suffers an indoubt failure, (the transaction must be associated with a logically recoverable queue) another trigger transaction cannot be attached until the indoubt failure has been resolved.

Transient data services

The following services are performed by the transient data program in response to transient data commands issued in application programs:

`Intrapartition data disposition`
Controls and queues data for serially reusable or re-enterable facilities (programs, terminals) related to this partition or region.

`Intrapartition data acquisition`
Retrieves data that has been placed in a queue for subsequent internal processing.

Extrapartition data acquisition
Enters a sequentially organized data set into the system.

Extrapartition data disposition
Writes fixed- or variable-length data in a blocked or unblocked format on sequential devices, usually for subsequent offline processing.

Automatic transaction initiation
Initiates a transaction to process previously queued transient data when a predefined trigger level is reached.

Dynamic open/close
Logically opens or closes specified extrapartition data sets (queues) during the real-time execution of CICS.

Dynamic allocation and deallocation of extrapartition queues
Extrapartition transient data queues do not have to be predefined in your JCL. They can be created dynamically.

Transient data

This section describes transient data's interfaces.

Intrapartition queues

Figure 101 shows transient data's interfaces for intrapartition queues.

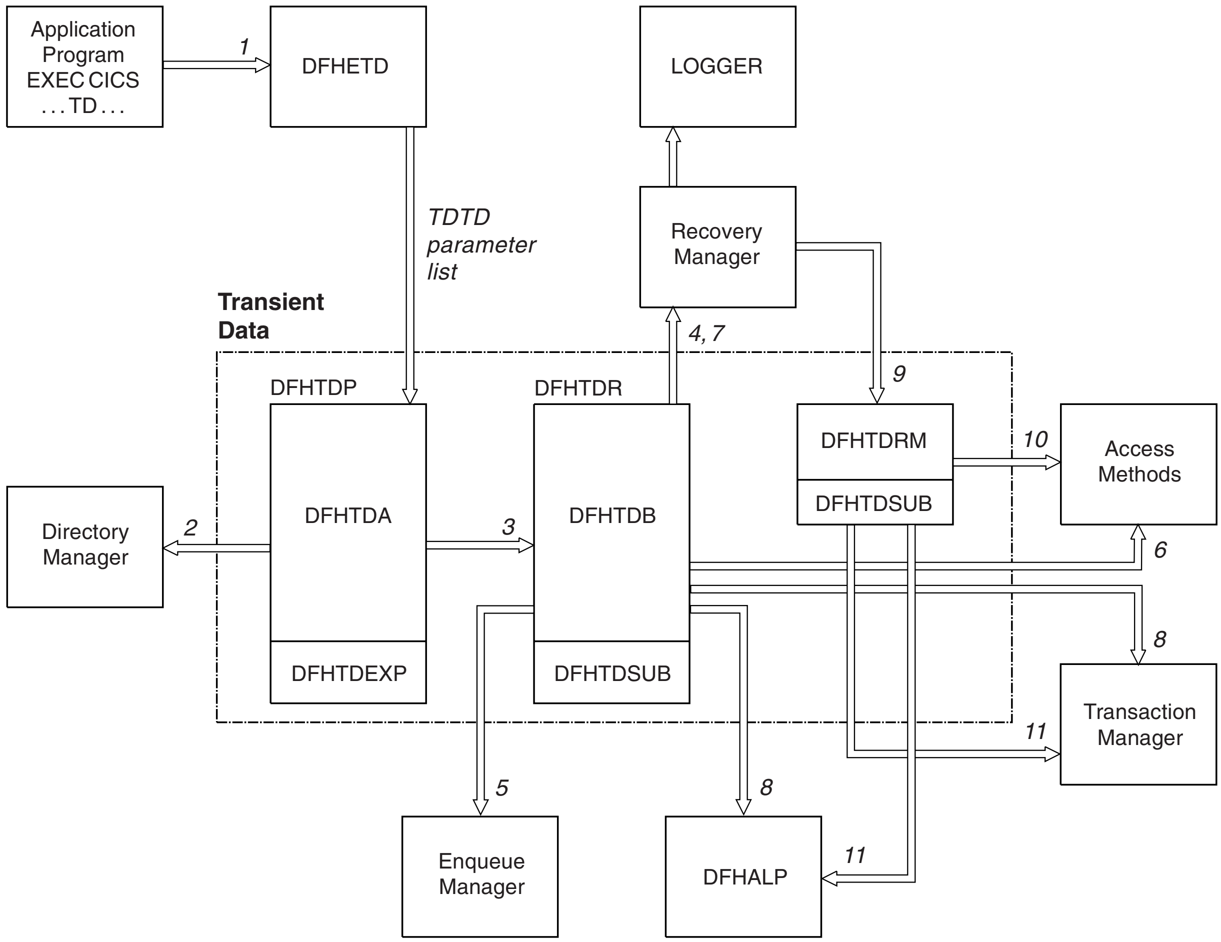


Figure 101. Transient data interfaces for intrapartition queues

Notes:

1. An application program invokes a Transient Data request (WRITEQ TD, READQ TD, or DELETEQ TD). The EXEC interface module, DFHETD is invoked and calls Transient Data using the TDTD CDURUN parameter list.
2. Transient Data locates the target queue using a Directory Manager locate.
3. Assuming that the required queue has been found, the call is passed to the module that handles intrapartition queue requests, DFHTDQ.
4. If the target queue is logically recoverable, Transient Data must tell Recovery Manager it is interested in this UOW by setting its work token in the Recovery Manager's table.
5. If the target queue is logically recoverable, Transient Data must obtain an enqueue on the appropriate end of the queue by invoking the Enqueue Manager.
6. Data is read from (or written to) the target queue using the appropriate access method. In the case of physically recoverable queues only, the buffers are always flushed and the data set hardened.
7. After the request has completed, Transient Data must log the state of the queue, if the queue is physically recoverable.
8. If the request was a WRITEQ TD request and the target queue was physically recoverable or non-recoverable, the trigger level may have been exceeded. If the trigger transaction is to be associated with a terminal DFHALP is invoked so that the required AID can be scheduled. If the trigger transaction is to be associated with a file, Transaction Manager is invoked to attach the trigger transaction.

9. If a UOW has updated a logically recoverable queue, Recovery Manager invokes Transient Data when the UOW begins syncpoint processing DFHTDRM.
10. Transient Data invokes the appropriate access methods to harden the data set. Finally, Recovery Manager invokes Transient Data once more, detailing whether Transient Data should commit or back out its updates.
11. If the UOW commits the updates. Transient Data attaches a trigger transaction or schedules an AID if the trigger level has been exceeded. DFHALP is invoked if the trigger transaction is associated with a terminal. Transaction Manager is invoked if the trigger transaction is associated with a file.

Extrapartition queues

Figure 102 shows the transient data interfaces for extrapartition queues.

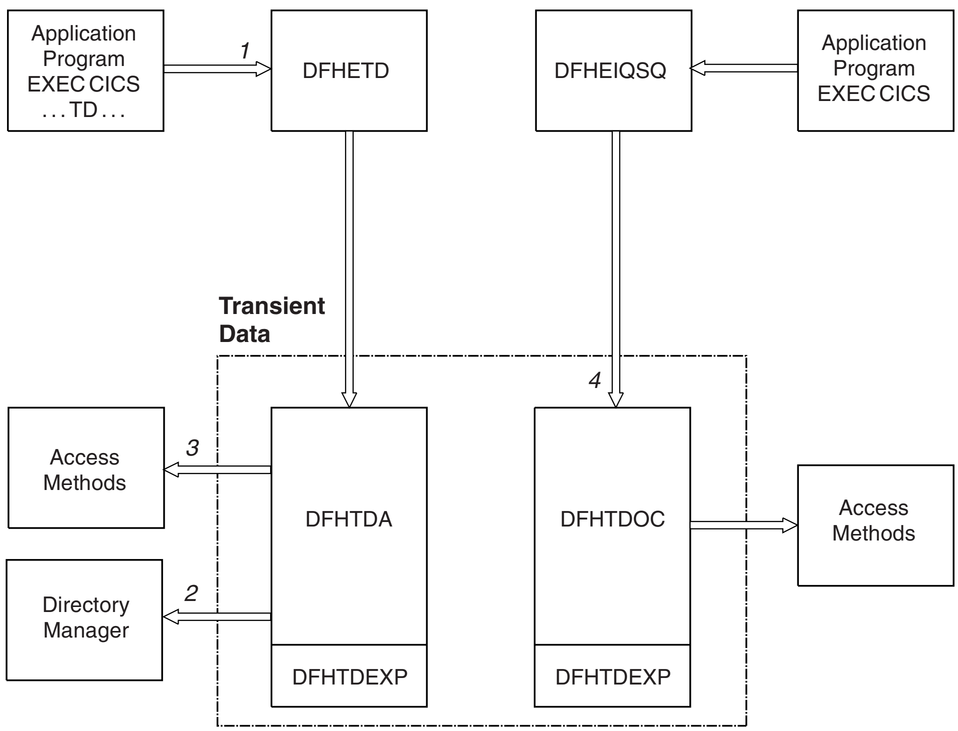


Figure 102. Transient data interfaces for extrapartition queues

Notes:

1. An application program invokes Transient Data services (WRITEQ TD, READQ TD or DELETEQ TD). The EXEC interface module, DFHETD is invoked. DFHETD invokes Transient Data using the TDTD CDURUN parameter list.
2. Transient Data locates the target queue using Directory Manager.
3. The request is passed to the appropriate QSAM routine for processing. QSAM PUT with LOCATE mode is used.
4. If an application program requests that an intrapartition queue be opened or closed, module DFHTDOC is invoked using the TDOC CDURUN parameter list.

Modules

Module	Function
DFHTDP	Provides request analysis and extrapartition processing, RMODE(24)

Transient data control

Module	Function
DFHTDA	Included in load module DFHTDP. Provides request analysis and processing for extrapartition queues
DFHTDEXC	Included in load module DFHTDP. Contains subroutines associated with the processing of extrapartition queues
DFHTDOC	Included in load module DFHTDP. Manages the opening and closing of extrapartition queues
DFHETD	Processes EXEC CICS commands and maps them to the TDTD CDURUN parameter list
DFHTDB	Included in load module DFHTDQ. Processes intrapartition queue requests
DFHTDSUC	Included in load module DFHTDQ. Contains subroutines associated with the processing of intrapartition transient data queues
DFHTDRM	Undertakes syncpoint processing on behalf of Transient Data
DFHTDTM	Manages requests to install, discard, set and inquire on transient data queues

Exits

The following global user exit points are provided for this function: XTDREQ, XTDEREQ, XTDEREQC, XTDIN, and XTDOUT.

See the *CICS Customization Guide* for further information.

Trace

The following point ID is provided for transient data control:
- AP F6xx, for which the trace levels are TD 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 64. User exit control

User exit control enables the user to run exit programs at selected points in CICS modules in the application domain and in other domains. The exit program can be enabled or disabled dynamically, and useful information can be transferred to a user work area.

This function:

- Controls which exit programs are to run at which exit points. This is generally specified using EXEC CICS commands and can be changed during a CICS run.
- Invokes the specified exit programs when control reaches an exit point in a CICS module, and handles any change in flow indicated by a return code from the user exit program.

Design overview

User exit control provides an interface that allows the user to run exit programs at selected points (known as exit points) in CICS control modules. The exit programs are separate from the control modules and are associated with them dynamically by means of the EXEC CICS ENABLE command. (See the *CICS Customization Guide* for a description of how to use exit programs.)

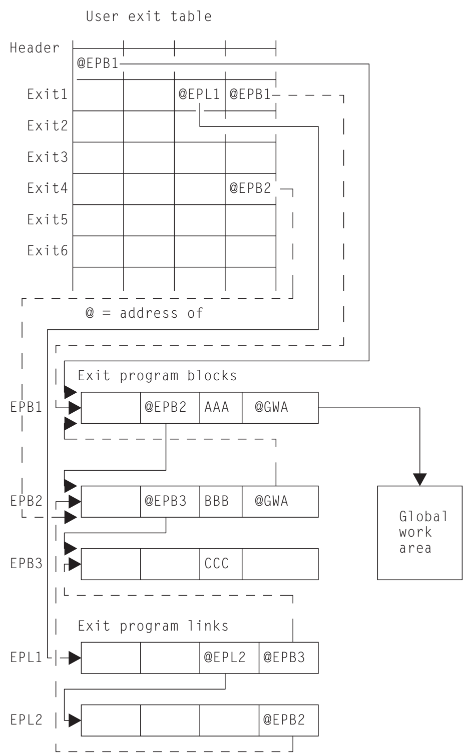
An exit point can have more than one exit program, and an exit program can be shared by more than one exit point. Work areas can be set up for the exit programs, and several exit programs can share a work area. For some exit points, the continuation of the control module can be controlled by a return code.

Each exit point is identified internally by an exit number. The user exit table (UET) contains a UET header and an entry for each exit point, in exit-number order. The UET is addressed from CSAUETBA in the CSA and exists throughout the life of CICS.

Each enabled exit program is represented by an exit program block (EPB). This exists only while an exit program is enabled or while any other exit program is using the work area owned by this exit program. The EPBs are chained together in order of enablement. The UET header points to the first EPB.

Each activation of an exit program for a particular exit point is represented by an exit program link (EPL) which points to the EPB for the exit program. The first EPL for each exit point is contained in the UET entry. If an exit point has more than one exit program, additional EPLs are obtained to represent each subsequent activation. These additional EPLs are chained off the UET entry in order of activation. Thus, for each exit, its EPL chain defines the exit programs that are to be executed at that exit point, and the order of execution.

The user exit interface (UEI) control blocks are illustrated in Figure 103.



Notes:

1. There are three enabled programs: AAA, BBB, and CCC.
2. Program AAA owns a global work area, which is shared by program BBB. The global work area pointer (@GWA) in BBB's EPB points to the EPB of the program owning the shared area, namely AAA's EPB.
3. Exits 1 and 4 are associated with these exit programs.
4. For Exit 1, exit programs AAA, CCC, and BBB have been activated, in that order, as indicated by the EPL chain.
5. Exit program BBB has been activated for exit 4.

Figure 103. UEI control blocks

All user exit programs are executed in the AP domain. When exit programs are activated for exit points in other domains, control is passed from the domain to the AP domain's user exit service module, which creates the necessary environment to invoke the exit programs via the user exit subroutine.

User exit control modules

This section describes the function of the user exit control modules.

DFHUEM (user exit manager)

The user exit manager (DFHUEM) processes EXEC commands that are entered by an application program or the command interpreter to control user exit activity. DFHUEM contains three routines, corresponding to the three commands, as follows:

ENABLE
: Checks whether an EPB already exists for the exit program specified in the PROGRAM operand.
  - If an EPB is not found and the ENTRY operand is not specified, the exit program is loaded, and:
    1. A new EPB is obtained and added to the chain.
    2. The name and entry address of the exit program are placed in the EPB.

3. If the GALENGTH operand is specified, a work area is obtained, and its address and length are placed in the EPB.
4. If the GAPROGRAM operand is specified, the address of the EPB for the exit program specified in the GAPROGRAM operand is placed in the new EPB, thus allowing exit programs to share a global work area.

- If the EXIT operand is specified, the EPL chain for the specified exit point is found.
  1. A new EPL is obtained, if necessary, and added to the chain.
  2. The address of the EPB for the exit program specified in the PROGRAM operand is placed in the EPB.
  3. The activation count in the EPB is increased by 1.
  4. If the exit point is not in the AP domain, the domain is notified that the exit point is active.
- If the START operand is specified, the start flag in the EPB is set on.

DISABLE
Finds the EPB for the exit program specified in the PROGRAM operand.

- If the STOP or EXITALL operand is specified, the start-flag in the EPB is set off.
- If the EXIT operand is specified, the EPL chain for the specified exit point is found. The EPL pointing to the EPB for the exit program specified in the PROGRAM operand is removed from the chain and the activation count is reduced by 1.
- If the EXITALL operand is specified:
  1. All EPL chains are scanned.
  2. All EPLs pointing to the EPB for the exit program specified in the PROGRAM operand are removed from its chain.
  3. If the ENTRY operand was not specified when the exit program was enabled, the exit program is deleted.
  4. The EPB is removed from the chain.
  5. If a work area used by the exit program is not still being used by another exit program, it is released.
  6. Any EPB or EPL that is no longer required is moved to a free-chain anchored in the UETH.
- When EXIT or EXITALL is specified for exit points not in the AP domain, the domain is notified when there are no exit programs active.

EXTRACT-EXIT
Finds the EPB for the exit program specified in the PROGRAM operand. The work area's address and length are extracted from this EPB (or from the EPB that owns the work area) and placed in the user's fields specified in the GASET and GALENGTH operands.

DFHUEH (user exit handler)

The user exit handler module, DFHUEH, is used to process exit points in the AP domain.

At each exit in a control module, there is a branch to the DFHUEH program. This module scans the EPL chain for that exit and invokes each started exit program in the chain, passing it a parameter list and a register save area. On return from each exit program, the return code is checked and a current return code (maintained by DFHUEH for return to the control module) is set as appropriate.

DFHAPEX (user exit service module)

The user exit service module, DFHAPEX, is used to process exit points in domains other than the AP domain.

When an exit point is reached in a non-AP domain, control is passed to the user exit service module (DFHAPEX) in the AP domain, if the domain has previously been notified that there is an exit program activated for the exit point.

User exit control

The user exit service module constructs the user exit parameter list, using special parameters from the domain, and invokes the user exit subroutine (DFHSUEX).

The return code from DFHSUEX is passed back to the calling domain.

DFHSUEX (user exit subroutine)

The DFHSUEX module invokes all started user exit programs for an exit point in a domain (other than the AP domain) by scanning the EPL chain, using the same processing as the user exit handler (DFHUEH). The parameter list defined by DFHAPEX is passed to the exit programs. Return codes from the exit programs are checked and returned to DFHAPEX.

Control blocks

The control blocks associated with the user exit interface are illustrated in Figure 104 and listed below. Further information about the control blocks is given in the “Design overview” on page 467 and in Figure 103 on page 468.

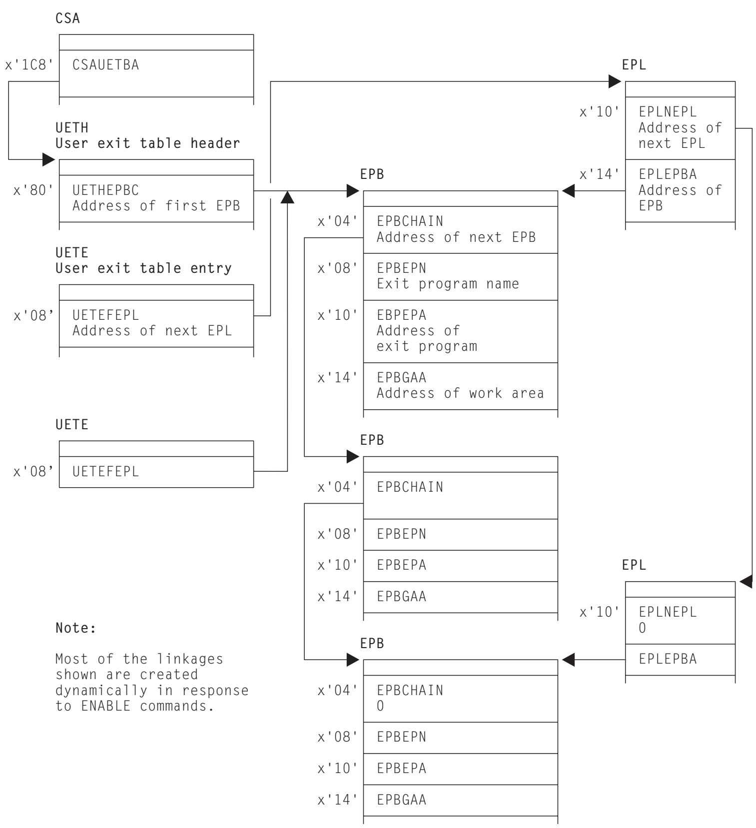


Figure 104. Control blocks associated with the user exit interface

The main control blocks are as follows:

UETH User exit table header

UETE User exit table entry—one for every exit point

EPB Exit program block—one for every enabled user exit program, containing information about the location and activity of the program, and any global work area owned or shared by the program

EPL. Exit program link—each EPL indicates one exit program to be invoked at an exit point and which EPL, if any, contains information about the next program to be invoked at that exit point.

See the *CICS Data Areas* manual for a detailed description of these control blocks.

Modules

Module	Function
DFHAPEX	The interface between an exit point in a domain (other than the AP domain) and the AP domain.
DFHSUEX	Handles the invocation of user exit programs at exit points in CICS domains (other than the AP domain). Processing is similar to DFHUEH, passing a parameter list defined in DFHAPEX.
DFHUEH	Links an exit point in a CICS management module in the AP domain and the user code. DFHUEH invokes in turn each started exit program for that exit point, passing a parameter list defined in the CICS management module.
DFHUEM	The EXEC interface processor for the ENABLE, DISABLE, and EXTRACT user exit commands.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:

- AP D5xx, for which the trace levels are UE 1, AP 1, AP 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

For user exit programs running at an exit point within the AP domain, UE level-1 trace entries are produced.

For user exit programs running at an exit point in a CICS domain other than the AP domain, the UE level-1 trace entries are not produced. Instead, the D5xx trace entries for AP level 1 and AP level 2 are available, providing more information than the UE trace. For AP level 1, the DFHUEPAR parameter list is traced, containing the addresses of fields special to that exit point. For AP level-2 tracing, the contents of the fields are printed, each field being truncated to 200 bytes if necessary.

Chapter 65. VTAM generic resource

This section describes how the generic resource support provided by release 4.2 of VTAM is used by CICS.

A CICS system may register as a VTAM generic resource. It may then be known either by its unique applid or by the generic resource name which is shared by a number of CICS systems, all of which are registered to the same generic resource.

For more information about CICS support for VTAM generic resource consult the CICS Release Guide or the CICS Intercommunication Guide. Consult the VTAM Programming Manual for information about generic resource from the VTAM point of view.

Design Overview

If CICS is to register as a generic resource member, the GRNAME system initialization parameter must be specified.

If GRNAME is specified CICS attempts to register immediately after the ACB is open by issuing the VTAM SETLOGON OPTCD=GNAMEADD command.

If registration succeeds, CICS is then a member of the generic resource specified by the SIT GRNAME parameter and may be addressed either by its generic resource name or (subject to certain restrictions) by its unique applid. Use of the generic resource name allows VTAM to balance the workload by selecting whichever generic resource member is most lightly loaded.

If registration fails, CICS initialization continues but CICS will not be a generic resource member.

The registration status may be examined by means of the CEMT INQUIRE VTAM command.

CICS de-registers as a generic resource by means of the VTAM SETLOGON OPTCD=GNAMEDEL command immediately before the ACB is closed.

Generic resource and LU6.1/LU6.2

Although terminals may log on freely using either the generic resource name or the member name this is not the case with LU6.1 and LU6.2 connections which are more restricted in their use of member names.

LU6.2 GR to GR connections

For LU6.2 connections between generic resources the design makes use of LU6.2 autoinstall. Only connections which are intended to issue an ACQUIRE need be defined and these must all have the generic resource name specified as the NETNAME.

Two types of connection are possible.

- Generic resource name connections. These are connections which have the generic resource name as the NETNAME. NETNAMEs must be unique and so there can only be one of these per partner generic resource.
- Member name connections. These are connections which have the unique applid (member name) as the NETNAME.

Since there can only be one generic resource name connection for each partner generic resource it follows that most connections will be member name connections.

VTAM generic resource

EXEC CICS INQUIRE CONNECTION or CEMT INQUIRE CONNECTION may be used to determine which is the generic resource name and which the member name.

When the first BIND from a different generic resource comes into the SCIP exit (DFHZBLX), a generic resource name connection will be established. If no predefined generic resource name connection exists one will be autoinstalled. Subsequent BINDs coming into DFHZBLX from different members of the same generic resource will cause member name connections to be autoinstalled. A member name connection should never be defined for a member of a different generic resource because this creates the possibility of having two definitions (TCSE's) for the same connected system.

Communications between members of the same generic resource must be by member names only.

Two new bits TCSE_GR and TCSE_GRNAME_CONN have been introduced to indicate the different connection types. They are only valid for LU6.2 connections between generic resources.

The table shows different values of TCTENNAM, TCSESID and TCSEX62N for LU6.2 connections between generic resources, depending on the settings of TCSE_GR and TCSE_GRNAME_CONN:

TCSE_GR TCSE_GRNAME_CONN	ON ON	ON OFF
TCTENNAM TCSESID TCSEX62N	GRname GRname membername	membername membername GRname

LU6.2 GR to non-GR connections

If a single (non-generic resource) system has an LU6.2 connection to a generic resource member it may use either the generic resource name or the member name as the NETNAME.

If the member name is used the initial acquire of the connection must be done by the non-generic resource partner. This means that the generic resource side must not have autoconnect set on. This is because the generic resource partner relies on VTAM to tell it if it is to known by its member name. VTAM does this by setting a bit which is valid for the first BIND only. Sessions can be acquired by either partner once the SNASVCMG sessions have bound.

For these connections TCSE_GR is always set off and TCSE_GRNAME_CONN has no meaning on both systems. The rule here is that TCSESID always contains the NETNAME (as defined in the RDO connection definition) and TCSEX62N always contains the member name (unique applid). The table illustrates this:

TCSE_GR TCSE_GRNAME_CONN RDO_HOSTNAME	OFF not applicable GRname	OFF not applicable membername
TCTENNAM TCSESID TCSEX62N	GRname GRname membername	membername membername membername

If the generic resource name is to be used, the single system may itself be made into a generic resource allowing it to exploit the design for communications between generic resources. If this is not possible the solution is to use a "hub" or code a generic resource resolution exit to ensure that not more than one member of a generic resource communicates with the single system at any one time using the generic resource name. (The use of "hubs" is described in the CICS Intercommunications Guide).

LU6.1

There is no autoinstall for LU6.1, and so less flexibility is allowed for LU6.1 connections between generic resources. CICS-CICS LU6.1 connections can only communicate by generic resource names and must use a "hub" or a generic resource resolution exit.

TCSE_GR and TCSE_GRNAME_CONN do not apply to LU6.1. For LU6.1 connections with a generic resource the generic resource name is in TCTENNAM and TCSESID and the member name is in TCSEX61N.

Ending affinities

Affinities are records held by VTAM to show it where to direct data flows within a generic resource. Some of these affinities are "owned" by CICS. These are affinities for LU6.2 synclevel 2, LU6.2 limited resources and LU6.1 connections. They may be ended by means of the SET CONNECTION ENDAFFINITY and PERFORM ENDAFFINITY commands.

Generic resource and ATI

This section applies only to those terminals which are logged on using the generic resource name.

When an ATI request is issued in an AOR for a terminal that is logged on to a TOR, CICS uses the terminal definition in the AOR to determine the identity of the TOR to which the request should be shipped. If there is no terminal definition in the AOR, the “terminal-not-known” global user exits (XICTENF and XALTENF) may be used to supply the name of the TOR.

However, if the TOR in question is a member of a generic resource and the user has logged on using the generic resource name, VTAM will have connected the terminal to the generic resource member which was most lightly loaded at the time. If the user then logs off and on again the terminal may be connected to a different generic resource member. If this happens, the TOR which is to receive the ATI request cannot be determined from the terminal definition in the AOR or the “terminal-not-known” user exit.

CICS solves the problem in the following manner:

1. The ATI request is first shipped to the TOR specified in the terminal definition in the AOR (or by the “terminal-not-known” exit). If the terminal is logged on to this TOR (the “first-choice” TOR) the ATI request completes as normal.
2. If the terminal is not logged on to the first-choice TOR, the TOR issues a VTAM INQUIRE OPTCODE=SESSNAME to find which generic resource member, if any, the terminal is now logged on to. This information is passed back to the AOR and the request is then shipped to the correct TOR.
3. If the first-choice TOR is not available, the AOR issues a VTAM INQUIRE OPTCODE=SESSNAME to find where the terminal is now logged on. The INQUIRE is not attempted in the following situations:
   - The VTAM in the AOR is a pre-4.2 version and does not support generic resource.
   - The AOR was started with the VTAM system initialization parameter set to NO.

   The INQUIRE will not succeed if the TORs and the AOR are in different networks.

   If the INQUIRE is successful the ATL request is shipped to the TOR where the terminal is logged on.

Modules

DFHZBLX

DFHZBLX is a new module which has been created to deal with LU6.2 BIND processing. Part of its function was formerly part of DFHZSCX. It is link-edited with DFHZSCX and is still logically part of it, but it returns directly to VTAM, not via DFHZSCX.

VTAM generic resource

There is a new part of the module, apart from that which was once contained in DFHZSCX, which deals with generic resource BIND processing. If CICS is registered as a generic resource and the partner is also a generic resource, DFHZBLX has to decide on the appropriate type of connection. This may be either a generic resource name connection, in which the NETNAME is the partner's generic resource name, or a member name connection, in which the NETNAME is the partner's member name.

DFHZBLX is also responsible for setting the bits in the connection entry which are specific to generic resource.

If CICS is not registered as a generic resource, the generic resource code is not invoked.

DFHZGCH

DFHZGCH is a domain subroutine which is called by DFHEIQSC after one of the following commands.

EXEC CICS SET CONNECTION ENDAFFINITY

CEMT SET CONNECTION ENDAFFINITY

EXEC CICS PERFORM ENDAFFINITY

CEMT PERFORM ENDAFFINITY

Its function is to issue the VTAM CHANGE OPTCD=ENDAFFINITY command.

If the affinity is ended successfully,

the connection is deleted if it is autoinstalled.

If the connection is defined,

the generic resource specific information in the connection entry is reset,

the catalog entry is updated,

the connection is deleted from the TCSM index.

The VTAM return codes are reflected back to DFHEIQSC.

DFHZGIN

DFHZGIN is a domain subroutine.

In a TOR it is called by DFHCRS when a request has been shipped from a remote system, if a terminal cannot be located.

In an AOR it is called by DFHALP when the schedule of an AID fails because the TOR has gone away.

It has two functions:

1. INQUIRE_NQN

   A VTAM INQUIRE OPTCD=NQN is issued to find the fully qualified NETNAME of a terminal given the NETNAME as input. The fully qualified NETNAME is required for INQUIRE OPTCD=SESSNAME.
2. INQUIRE_SESSNAME

   A VTAM INQUIRE OPTCD=SESSNAME is issued to find which member of a generic resource a terminal is logged on to given a fully qualified NETNAME as input.

The following responses are returned to the caller:

- OK - VTAM return code was X'00' fdb2 X'00'
- NOT FOUND - VTAM return code X'14' fdb2 X'88'
- EXCEPTION - The call was rejected for some other reason than not found.

For the exception case an exception trace is written and a message in the range DFHZC0182 - DFHZC0185 is output to the CSNE log giving the VTAM return codes.

Problem solving for generic resource

Trace TC level 1, 2 & exception in the ranges AP FA50-FA59, FAB0-FABA and FB87-FB8F.

Messages DFHZC0170 to DFHZC0185 are written to the console and CSNE logs.

Information output by DFHZNAC following BIND failures.

If a dump is produced examine the generic resource status and generic resource flag bytes.

The following symptoms may indicate that an affinity should be ended and has not been.

- Sessions failing to acquire with message DFHZC2405 "Node not activated". This may also indicate a setup error.
- Sessions failing to acquire with various instances of DFHZC2411. This may also indicate that a rule has been violated.
- CICS fails to register as a generic resource when it has previously been a member of a different generic resource. Message DFHZC0171 is written to the console with VTAM rtncd X'14' fdb2 X'86'.
- Connections autoinstalling unexpectedly. If a non-generic resource is addressing a generic resource member by its member name this may also indicate that the first ACQUIRE was issued from the generic resource side.

Generic resource status byte (TCTV_GRSTATUS)

TCTV_GR_REGD (X'80')
This CICS is registered as a member of a generic resource.

TCTV_GR_REGERR (X'40')
This CICS attempted to register as a generic resource member (SIT GRNAME parameter specified) but the attempt was rejected by VTAM.

TCTV_GR_NOTAVAIL (X'20')
This CICS attempted to register as a generic resource member (SIT GRNAME parameter specified) but the level of VTAM was not 4.2 or above.

TCTV_GR_DREGD (X'08')
This CICS was previously a member of a generic resource but has successfully de-registered.

TCTV_GR_DREGERR (X'04')
This CICS attempted to de-register as a member of a generic resource by issuing SETLOGON OPTCD=GNAMEDEL but the attempt was rejected by VTAM.

TCTV_GR_NOTAPPL (X'02')
The GRNAME system initialization parameter was not specified.

TCTV_GR_NOTREG (X'00')
CICS is not registered as a generic resource and has not attempted to register. (Holds this value before registration is attempted, if required.)

Generic resource flag byte (TCSEI_GR)

TCSE_GR (X'80)
Both partners are registered as generic resources. Valid from initial acquire to ENDAFFINITY.

TCSE_GR_NAME_CONN (X'40')
Set on for a generic resource name connection in which TCSESID contains the generic resource name and TCSEX62N contains the member name.

Set off for a member name connection in which TCSESID contains the member name and TCSEX62N contains the generic resource name.

This bit is only meaningful if TCSE_GR is set on.

TCSE_USE_OUR_MEMBER_NAME (X'20')
The partner is using our member name. (An indication that the member name, not the generic resource name must be passed in the BIND).

TCSE_MSG179_ISSUED (X'10')
Message DFHZC0179 has been issued. This message is issued when the secondary SNASVCMG session binds if TCSE_GR is set. It makes clear which is the generic resource name and which the member name of the partner session.

TCSE_CATLG_DONE (X'08')
A defined connection with an affinity has been catalogued.

TCSE_MSG177_ISSUED (X'04')
Message DFHZC0177 has been issued. This message is output whenever an LU6.2 limited resources, LU6.2 synclevel 2 or LU6.1 connection is acquired. It is output when the secondary SNASVCMG session binds. It is intended to alert the user to the fact that acquiring the connection has caused an affinity to be created and gives the NETNAME and NETID of the partner.

Trace

Trace point ids

- FA50 - FA59

  are provided for problem determination during ENDAFFINITY processing. (Module DFHZGCH)
- FAB0 - FABA

  are provided for problem determination during INQUIRE SESSNAME processing. (Module DFHZGIN)
- FB87 - FB8F

  are provided for problem determination during generic resource registration and de-registration. (Module DFHZGSL)

Waits

Module	Type	Resource Name	Resource Type	ECB	Function
DFHZGCH	MVS	CHANGECB	ZC_ZGCH	CHANGECB	Wait for completion of INQUIRE SESSNAME
DFHZGIN	MVS	INQ_ECB	ZC_ZGIN	INQ_ECB	Wait for ENDAFFINITY to complete

Chapter 66. VTAM LU6.2

This section describes the layer of CICS that manages the interface to VTAM for LU6.2 communication. VTAM LU6.2 provides advanced program-to-program communication (APPC) between transaction-processing systems, and enables device-level products (APPC terminals) to communicate with host-level products and with each other. APPC sessions can therefore be used for CICS-to-CICS communication, and for communication between CICS and other APPC systems (for example, AS/400®) or terminals.

For information about the CICS functions that you can use to exploit LU6.2 communication, see Chapter 13, “Distributed program link,” on page 107, Chapter 14, “Distributed transaction processing,” on page 109, Chapter 26, “Function shipping,” on page 277, Chapter 29, “Intersystem communication (ISC),” on page 305, Chapter 62, “Transaction routing,” on page 441.

Design overview

The main feature that distinguishes LU6.2 from other LU types is the support for parallel sessions i.e. many sessions (and conversations) between the two LUs at the same time. These sessions are further grouped by use of the class of service facility in VTAM. The TCT structure for LU6.2 reflects this. Under the system entry (TCTSE) are a series of mode group entries (TCTMEs). Within a mode group there are a number of sessions represented by terminal entries (TCTTEs).

All the sessions within a mode group have the same transmission characteristics, that is, the same class of service. When a request to ALLOCATE a session is made, a MODENAME can be specified, indicating which class of service is required.

When a session has been allocated and a conversation started, data can be received and sent between the connected LUs. This is more or less directly under the control of the CICS application in the case of DTP, or indirectly under the control of the user for the other ISC facilities.

CICS also supports LU6.2 single session connections. These are represented by a TCTSE, a single TCTME and a single TCTTE. They support the same functions as parallel session connections.

Detailed information about VTAM LU6.2 commands and macros is given in the relevant VTAM manuals.

Session management

Systems Network Architecture (SNA) defines several processes to be used in managing LU6.2 sessions. The CICS implementation provides transaction code for the following Transaction Program Names (TPNs) defined by LU6.2.

- X'06F1' = CHANGE_NUMBER_OF_SESSIONS (CNOS)
- X'06F2' = EXCHANGE_LOG_NAME (XLN)

The required transaction definitions are:

TRANSACTION	XTRANID	PROGRAM
CLS1	X'06F10000'	DFHZLS1
CLS2	X'06F20000'	DFHCLS3

These resource definitions are provided in the DFHISC group.

So that the SNA service transaction programs can always communicate with each other, even when all the sessions between two systems are busy, two extra sessions are always created whenever parallel sessions exist between two systems. CICS generates these two extra sessions (with a reserved MODENAME of SNASVCMG) unless SINGLESESS(YES) is specified for the connection. Only SNA service transaction programs are allowed to use these two sessions.

Change Number Of Sessions (CNOS)

When there are parallel sessions between two LU6.2 systems, it is possible to vary the number of sessions available using CEMT or EXEC CICS commands, either for the entire connection, or by modegroup. The number of available sessions for a modegroup is called the SESSION LIMIT. It corresponds to the number of in-service sessions in that modegroup. The two systems must agree on the session limit for a modegroup at any given time. To achieve this, the LU6.2 architecture defines a CNOS service transaction program which runs in each system, communicating with its counterpart using architected CNOS commands and replies. They negotiate the session limit and the numbers of contention winners and losers at each end. For CICS, the CNOS service transaction program is DFHZLS1.

CNOS commands are not required for the SNASVCMG modegroup on parallel session connections, or for single session connections, because the session limits are fixed.

Figure 105 shows the flow of control for CNOS operations.

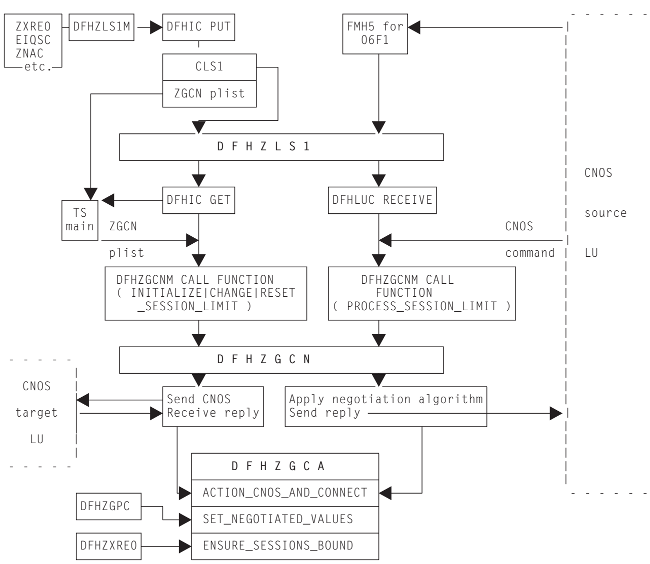


Figure 105. Flow of control for CNOS

Exchange Log Name (XLN)

When DFHZNAC determines that it is necessary to exchange log names with a remote system, it starts the syncpoint resynchronization transaction, using the DFHCRERI macro specifying FUNCTION(XLN). The main program for this transaction is DFHCRRSY (in load module DFHLUP). When DFHCRRSY determines that resynchronization is required it will schedule other instances of itself to perform the resynchronization.

When TPN X'06F2' is received from a remote system, DFHCRRSY is called to handle the inbound Exchange Log Names and resynchronization.

LU6.2 session states

The following CICS modules maintain specific states of LU6.2 sessions.

Module	State	Macro
DFHZBKT	SNA bracket state	DFHZBSM
DFHZCNT	Contention state	DFHZCNM
DFHZCHS	Chain state	DFHZCHM
DFHZCRT	RPL_B state	DFHZCRM

These modules are invoked via the macros shown in the last column. Any query or change to the states is performed using these macros.

The LU6.2 states for each session are stored in the TCTTE for that session. The modules and associated TCTTE field are usually referred to as **state machines**. When a module, such as DFHZARL, wants to check that the session is in a suitable state to perform a given operation, it uses the appropriate state machine to perform the check by invoking the CHECK function of the relevant macro. If the operation subsequently causes a change in the state of the session, the SET function of the relevant macro is invoked to record the new state.

LU6.2 SEND and RECEIVE processing

LU6.2 SEND processing is done by DFHZSDL, using POST=SCHED to drive the VTAM exit DFHZSLX asynchronously when the request has been passed to VTAM.

DFHZRVL does LU6.2 RECEIVE processing, issuing the request to VTAM for asynchronous processing which drives the VTAM exit DFHZRLX on completion. DFHZRLX queues completed RPLs for further processing by DFHZRLP to a chain anchored off TCTVRPLQ in the TCT prefix. Entries are removed from the queue by DFHZDSP, and passed to the program designated to process the completed RPL. When authorized path VTAM support is used, the SEND and RECEIVE requests use the CICS high performance option (HPO) routines.

SEND and RECEIVE processing for LU6.2 use different RPLs:

- RECEIVE uses the receive RPL (also known as RPL_B, and addressed by TCTERPLB in the TCTTE LUC extension).
- SEND uses the send RPL (addressed by TCTERPLA in the TCTTE).

There are two exceptions when a SEND uses the receive RPL instead of the send RPL:

1. DFHZSDL sending a response
2. DFHZRLP sending DR1 response via synchronous SEND.

The processing state of the receive RPL is maintained in the LU6.2 RPL_B state machine field (TCTERPBS in the TCTTE LUC extension) by the DFHZCRT module and DFHZCRM macro combination, thus allowing rapid identification of the stage and type of RECEIVE being processed.

LU6.2 state machine transitions for contention, bracket, and chain states are performed via the DFHZCNM, DFHZBSM, and DFHZCHM macros as part of SEND and RECEIVE processing for LU6.2 sessions.

Limited resources

For efficient use of some network resources (for example, switched lines), SNA allows for such resources to be defined in the network as **limited resources**. Whenever a session is bound, VTAM indicates to CICS whether the bind is over a limited resource. Both single and parallel sessions may use limited resources.

The limited resources (LR) function is part of the LU6.2 base option set. When communicating over switched lines, it may be important to stop using this expensive resource as soon as possible. LR provides this facility. A bit in the BIND image is copied into the TCTTE to indicate LR usage. This bit (TCTE_LR) is used to determine whether CICS should UNBIND the link when the TCTTE is freed and no outstanding tasks are using the link.

SNASVCMG (parallel) sessions are not scheduled to be unbound until the initial CNOS exchange has been performed for all mode groups in the connection. They are then treated in the same way as user sessions.

Two bits in the terminal control table are used to reflect LR: TCTE_LR in the terminal entry (TCTTE) and TCSE_LR in the system entry (TCTSE). The following table shows the meanings of the TCTE_LR bit (ON or OFF) in combination with the TCTENIS 'node now in session' bits (YES or NO).

TCTE_LR	TCTENIS	Meaning
ON	YES	Current session over LR
ON	NO	Previous session over LR
OFF	YES	Current session not LR
OFF	NO	Never bound, or previous session not LR

TCSE_LR (in the system entry) is set ON when the first LR session is bound, and OFF as a result of CNOS negotiation to release the connection. If TCSE_LR is ON and there are no bound sessions, the connection state is then 'available'.

Modules

The modules listed below handle the VTAM LU6.2 support in CICS.
Session management state machines
- DFHZBKT
- DFHZCHS
- DFHZCNT
- DFHZCRT

Send and Receive processing
- DFHZRLP
- DFHZRLX
- DFHZRVL
- DFHZSDL
- DFHZSLX

CNOS
- DFHZLS1
- DFHZGCN
- DFHZGCA

Persistent Verification
- DFHCLS3

XLN and Resynchronization
- DFHCRRSY

DFHZRVL

DFHZRVL is invoked to issue an LU6.2 receive specific request to receive:
- Data

- Commands
- Responses
- Purge to end-chain (used by DFHZERH to clear incoming data)
- A single RU.

Two broad categories of RECEIVE data are recognized by CICS; both are processed as RECEIVE_WAIT requests to VTAM:

1. RECEIVE_WAIT, where CICS waits until input is received from VTAM before returning control to the caller. This applies to all RECEIVE response and command requests, and to data requests where the minimum length to be received is greater than zero.
2. RECEIVE_IMMEDIATE, where CICS immediately returns control to the caller without waiting for VTAM to complete the request unless the data is already in the VTAM buffer, in which case it processes the data in the same way as for RECEIVE_WAIT before returning to the caller. This is requested via a minimum length of zero. It is used by the RECEIVE_IMMEDIATE call for the SAA communications interface, by a LOOK_AHEAD call, and in support of timely receipt of responses, ensuring earlier detection of an ISSUE_ERROR response from the partner LU.

The receive buffer is set up to receive the data, and the address of the receive exit DFHZRLX (driven on completion of the request) is stored into the receive RPL (RPL_B) before the RECEIVE macro is issued to VTAM. DFHZRVL is used by DFHZERH to determine the state of the session.

DFHZRLP

This module completes the LU6.2 receive specific processing for LU6.2 requests.

RECEIVE_IMMEDIATE requests are processed in two phases, that is, on two passes through DFHZRLP:

1. The RPL_B state machine (TCTERPBS) is set to indicate that the RECEIVE has been completed by VTAM; then the exit is taken from DFHZRLP.
2. This phase corresponds to the single phase used for processing RECEIVE_WAIT requests, that is, the requests are checked for successful completion, examined to determine whether data, a command, or a response has been received, and parameters indicating what has been received are then returned to the caller.

Data received

When data is received, DFHZRLP:

1. Sets the bracket and chain state machines, and returns indicators to DFHZARL according to the DFC flags received with the data:
   - Response type
   - CD
   - EC
   - CEB
   - FMH
2. If more data is required, DFHZRLP recalls DFHZRVL via the activate scan routine (DFHZACT) to reissue the RECEIVE, for example when:
   - End-chain has not yet been received, and there is still room in the receive buffer. If the minimum length requested has already been received, the type of RECEIVE is altered from RECEIVE_WAIT to RECEIVE_IMMEDIATE resulting in a READ_AHEAD call in anticipation of there being more data available, and any data already in the VTAM buffer is processed by DFHZRLP before returning to the caller.
   - The original request was for data, and what has been received and processed is a command (only LUSTAT or BIS can validly be processed by DFHZRLP).
3. Returns control to DFHZARL when:
   - Sufficient data has been received for a BUFFER or LL type request.
   - End-chain has been received because of CD, RQD2, or CEB.

- FMH has been received.
- The call was incomplete, but insufficient space remains in the receive buffer for further data.

If the data was received with RQD1, a response is sent synchronously by DFHZRLP using the receive RPL.

Command received

When a command is received, the actions of DFHZRLP depend on the command:

- For LUSTAT6 received, the command is treated as data. If BB is included, then an exception response is sent (sense X'0813' or X'0814').
- For BIS received, CLSDST is requested and the receive re-driven.

All other commands are incorrect.

Response received

When a response is received, DFHZRLP:

1. Carries out checks:
   - Does the sequence number match the number of the BB request?
   - If it is a definite response, was it expected?
   - If it is an exception response, was it a session-level error?
2. Sets the state machines.
3. Passes back the return code to the caller.

DFHZSDL

This module issues the SEND request to VTAM to transmit data, commands, and responses on LU6.2 sessions.

DFHZSDL transmits:

- Data from a send buffer or an application area
- The commands:
  - LUSTAT
  - RTR
  - BIS
- Responses.

Data transmission

If a SEND LAST command is issued, any outstanding completed receive RPL is first processed by queuing the TCTTE for RECEIVE processing by DFHZRLP, and any incomplete receive RPL is canceled via RESETSR.

For data transmission, DFHZSDL uses:

LMPEO
: Large message performance enhancement outbound. VTAM slices large messages into RUs.

BUFFLST
: Buffer list. VTAM accepts data from non-contiguous buffers.

USERRH
: User request header. The request header is passed in BUFFLST.

A maximum of two buffer list entries are used. The first buffer list entry addresses the data in the send buffer, and the second the data in the application area.

The request header is built in the first buffer list entry using parameters passed from DFHZARL. If an implicit send was requested, then CD, RQD2, and CEB are not checked. The first-in-chain (FIC) indicator is set after checking the chain state machine, and last-in-chain (LIC) is set whenever CD, RQD2, or CEB is included. Null data sent only-in-chain (OIC) is converted to an LUSTAT6 command. The address of the send exit DFHZSLX is stored in the send RPL, and the VTAM SEND macro is issued. On completion of the SEND request, the bracket and chain state machines are set according to the DFC indicators. These state machines are used extensively by DFHZERH to determine the state of the session before executing an error request.

Command transmission

The LUSTAT6 command is sent with:
- CEB to terminate the BIND_in_bracket state
- Null data for OIC
- CB, RQD1 to BID for bracket.

The RTR command requests BB after a BID request is rejected with sense code X'0814'.

The BIS command shows bracket termination before CLSDST.

On completion of the SEND request, the exit DFHZSLX is invoked. LUSTAT causes the bracket and chain state machines to be set as for normal data flow.

Response transmission

DFHZSDL transmits ER1 and DR2 responses. The sequence number associated with the response is that of the path information unit (PIU) that initiated the current bracket. DFHZSDL uses the receive RPL (RPL_B) to send responses thus ensuring that the RU is returned with the response, unless the response is an ISSUE_ERROR request, in which case the send RPL is used. The response is sent synchronously, and POST=SCHED is included in the VTAM command, so that an exit routine is not involved. On return from VTAM, DFHZSDL sets the bracket and chain state machines accordingly.

DFHZSLX

The DFHZSLX module is the VTAM exit that is driven on completion of a SEND request. If the request completed successfully, the bracket and chain state machines are set to show the new state of the session. If the SEND request was data DR1, DFHZRVL is invoked via DFHZACT to receive the response.

DFHZRLX

The DFHZRLX module is the VTAM exit that is scheduled on completion of an LU6.2 RECEIVE_SPECIFIC request. DFHZRLX queues the completed RPL to a chain anchored from TCTVRLPQ in the TCT prefix. DFHZDSP dequeues the RPLs for further processing by DFHZRLP.

DFHCLS3

In the local CICS system, DFHCLS3 is invoked using the DFHLUS macro, which issues a DFHIC TYPE=PUT macro to start the appropriate transaction (CLS3) with data recorded on temporary storage indicating the requested operation.

The DFHLUS operations can be:

SIGNOFF	Sign off a user on the other LU
TIMEOUT	Time out users.

The SIGNOFF and TIMEOUT operations apply to persistent verification signons only.

DFHCLS3 retrieves the temporary-storage record.

The SIGNOFF and TIMEOUT operations are performed directly by DFHCLS3. These operations are supported outbound only.

For SIGNOFF, DFHCLS3 is started by DFHZCUT when a user on the other LU must be signed off.

For TIMEOUT, DFHCLS3 is started by DFHZCUT during time-out processing of a **persistent verification signed-on-from list**, also known to CICS as a local userid table (LUIT).

DFHCLS3 performs the following actions:
1. Calls DFHZCUT to find a userid that needs to be timed out
2. Makes a sign-off call to the other LU
3. Calls DFHZCUT to remove the userid from the LUIT.

This sequence is repeated until there are no more userids to be timed out.

If DFHCLS3 abends during time-out processing, control passes to a SETXIT routine in DFHCLS3, which calls DFHZCUT to tidy up the relevant LUIT.

DFHZLS1

DFHZLS1 is the main program for the CICS implementation of the CNOS SNA service transaction. When acting as the initiator of a CNOS request (the CNOS source), it is invoked by the DFHZLS1M macro issuing a DFHIC TYPE=PUT for transaction id CLS1. The possible commands on the CNOS source system are:-

- INITIALIZE_SESSION_LIMIT

  Acquire the specified connection, using the MAXIMUM values from the RDO SESSIONS definitions (for the required session limit and number of winner sessions) on the CNOS command for each modegroup.
- CHANGE_SESSION_LIMIT

  Negotiate a change of the current session limit for a specified modegroup.
- RESET_SESSION_LIMIT

  Release the connection, negotiating all modegroups to a session limit of zero.

When acting as the receiver of a CNOS request (the CNOS target), DFHZLS1 is invoked by an attach FMH for TPN X'06F1' sent from the CNOS source system, which is not necessarily CICS. The CNOS command sent with the attach FMH requests changes to the sessions in specified modegroups. In SNA terms, DFHZLS1 is handling a PROCESS_SESSION_LIMIT command. It issues a DFHLUC RECEIVE for the CNOS GDS that contains the details of the required command.

DFHZLS1 passes the parameters for each of the above commands through to DFHZGCN, where the detailed processing takes place.

DFHZGCN

DFHZGCN is an AP domain subroutine. It handles the four architected CNOS functions, as described below.

INITIALIZE_SESSION_LIMIT

This is a two pass function in CICS. First time through, DFHZGCN initiates the bind of the SNASVCMG winner session and returns. The bind processing eventually causes the “session started” routine in DFHZNAC to run. This re-issues the DFHZLS1M INITIALIZE_SESSION_LIMIT request, and the CNOS negotiation can then take place.

DFHZGCN performs the following actions:
1. Does a ‘privileged’ allocate (for a SNASVCMG session).
2. Builds an attach header.
3. Completes the building of the CNOS command, using MAXIMUM values in the TCTME.
4. Issues a SEND INVITE WAIT.

5. Issues a RECEIVE LLID.
6. Analyzes the responses to the command; SNA decrees that the CNOS source must accept the values returned.
7. Calls DFHZGCA to action the new values.
8. Sends messages DFHZC4900 and DFHZC4901 as appropriate.
9. Frees the session.

The above steps are repeated for each user modegroup in the connection.

RESET_SESSION_LIMIT

A connection release request is passed via DFHZLS1 to DFHZGCN.

DFHZGCN performs the following actions:
1. Does a ‘privileged’ allocate.
2. Builds an attach header.
3. Completes the building of one CNOS command, setting MAX, WIN, and LOS values to zero, and mode names affected to ALL.
4. Issues SEND INVITE WAIT.
5. Issues RECEIVE LLID.
6. Analyzes the response to the command; the CNOS target must accept zero sessions (DRAIN can be changed from ALL to NONE).
7. Calls DFHZGCA to action the new values.
8. Sends message DFHZC4900.
9. Frees the session.

CHANGE_SESSION_LIMIT

DFHZLS1 is started from the EXEC API or CEMT via DFHEIQSM to change the session limit for a specific modegroup.

DFHZGCN performs the following actions:
1. Does a ‘privileged’ allocate.
2. Builds an attach header.
3. Completes the building of one CNOS command, setting MAX and WIN values.
4. Issues SEND INVITE WAIT.
5. Issues RECEIVE LLID.
6. Analyzes the responses to the command; SNA decrees that the CNOS source must accept the values returned.
7. Calls DFHZGCA to action the new values.
8. Sends messages DFHZC4900 and DFHZC4901 as appropriate.
9. Frees the session.

PROCESS_SESSION_LIMIT

DFHZLS1 is attached, and calls DFHZGCN.

DFHZGCN performs the following actions:
1. Addresses the CNOS command that DFHZLS1 passed.
2. For each mode group specified, determines whether the values for session limit, source contention winners and source contention losers are acceptable. If not, the values are adjusted (negotiated) according to rules laid down by SNA.
3. If this system is currently performing shutdown, negotiates down to session limit zero.

4. Calls DFHZGCA to action the new values.
5. Sends the CNOS reply containing the negotiated values.
6. Sends messages DFHZC4900 and DFHZC4901 as appropriate.

DFHZGCA

DFHZGCA is an AP domain subroutine. It has three separate functions, as described below.

ACTION_CNOS_AND_CONNECT

After a CNOS negotiation DFHZGCA is responsible for changing the state of a specified modegroup to reflect the new values. There are three types of action required.

1. Put sessions in/out of service for session limit increase/decrease.
2. Set sessions to winner/loser in line with negotiated values.
3. Bind/unbind sessions for session limit decrease, autoconnect processing or contention polarity switch.

SET_NEGOTIATED_VALUES

This function is used by DFHZGPC during persistent sessions restart to set the saved CNOS values in the modegroup without any binding/unbinding of sessions.

ENSURE_SESSIONS_BOUND

DFHZXRE0 invokes this function during persistent sessions restart because recovery processing can lead to LU6.2 sessions becoming unbound. It is important to ensure that they are re-bound in accordance with the autoconnect setting.

Exits

No global user exit points are provided for this function.

Trace

All of the above mentioned modules have entry and exit trace points. Several of them also have exception and level 2 trace points. All of these trace points are from the AP domain and have ids in the range FB00-FCFF.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 67. VTAM persistent sessions

This section describes how CICS handles VTAM persistent session support. It uses VTAM 3.4.1 persistent LU-LU session improvements to provide restart-in-place of a failed CICS without the need for network flows to re-bind CICS sessions.

Persistent sessions can either be Single Node Persistent Sessions (SNPS) or Multi Node Persistent Sessions (MNPS) depending on how your VTAM network is set up.

If CICS is to support MNPS then PSTYPE=MNPS must be specified in the SIT.

The following mainly describes SNPS. Sections are added where MNPS differs from SNPS.

For an overview of persistent sessions, and a comparison with XRF, see the *CICS Transaction Server for z/OS Release Guide*.

For an introduction to this topic from the VTAM point of view, see the *Advanced Communications Function for VTAM Programming* manual, SC31-6348.

Design overview

CICS support of persistent sessions includes the support of all LU-LU sessions except LU0 pipeline and LU6.1 sessions. CICS determines for how long the sessions should be retained from the PSDINT system initialization parameter. (This is a user-defined time interval.) If a failed CICS is restarted within this time, it can use the retained sessions immediately—there is no need for network flows to re-bind them.

This interval can be changed using the CEMT SET VTAM command, or the EXEC CICS SET VTAM command, but the changed interval is not stored in the CICS global catalog, and therefore is not restored on an emergency restart.

If CICS is terminated through CEMT PERFORM SHUTDOWN IMMEDIATE, or if CICS fails, VTAM holds CICS' sessions in "recovery pending" state.

During emergency restart, CICS restores those sessions pending recovery from the CICS global catalog and the CICS system log to an "in session" state. This happens when CICS opens its ACB.

Subsequent processing is LU dependent: cleanup and recovery for non-LU6 persistent sessions is similar to that for non-LU6 backup sessions under XRF. Cleanup and recovery for LU6.2 persistent sessions maintains the bound session when possible but there may be cases where it is necessary to unbind and re-bind the sessions, for example, where CICS fails during a session resynchronization.

The end user of a terminal sees different symptoms of a CICS failure following a restart, depending on whether VTAM persistent sessions, or XRF, are in use:

- If CICS is running without VTAM persistent sessions, or XRF, and fails, the user sees the VTAM logon panel followed by the "good morning" message (if AUTOCONNECT(YES) is specified for the TYPETERM resource definition).
- If CICS does have persistent sessions support and fails, the user perception is that CICS is hanging: the screen on display at the time of the failure remains until persistent session recovery is complete. After a successful CICS emergency restart, the recovery options defined (in RECOVOPTION) for the terminals or sessions take effect. If SYSDEFAULT is specified as the value for RECOVOPTION, the user can clear the screen and continue to enter CICS transids. If MESSAGE is specified for the RECOVNOTIFY attribute of the TYPETERM resource definition, the user is notified of the successful recovery.

If CICS does not restart within the specified interval, the sessions are unbound, as if there has been a CICS failure without persistent sessions support in the system.

Note: SNPS support does not retain LU-LU sessions after VTAM, MVS, or CEC failure. Nor are sessions retained after the following commands:
- SET VTAM FORCECLOSE
- SET VTAM IMMCLOSE
- SET VTAM CLOSED
- PERFORM SHUTDOWN
- VARY INACT ID=applid

MNPS differs from SNPS in that MNPS support retains LU-LU sessions after a VTAM and MVS failure. The sessions are also retained after:
- SET VTAM FORCECLOSE

Persistent Sessions Restart flow

The following describes the flow of control for:
1. The enabling of persistence
2. The sessions that persist at start up time
3. The sessions that persist during dynamic open.

Enabling of persistence

Summary:
1. VTAM ACB opened with PARM=PERSIST=YES
2. VTAM levels checked.
3. VTAM SETLOGON OPTCD=PERSIST or NPERSIST

More detail: Persistence is enabled by:
1. The VTAM ACB is opened with PARM=PERSIST=YES - specified in DFHTCTPX.
2. DFHZSLS calls DFHZGSL to issue SETLOGON OPTCD=PERSIST/NPERSIST.

   DFHZSLS copies 8 bytes of VTAM information into the TCT prefix. These bytes contain details of the VTAM level and the functions which it supports. Previous releases of CICS only copy 4 bytes of VTAM data.

   The use of persistent sessions is dependent upon the level of VTAM present being at least V3R4.1. This level of VTAM returns more function bit data to CICS than previous versions and supports the use of persistent sessions. Checks are made by CICS of the current VTAM level and the VTAM level against which the TCT was generated. If either level is not high enough, parameters relating to the use of persistent sessions are not used when macros are called.

Sessions that persist at start up time

Summary:
1. Task CGRP runs DFHZCGRP
2. DFHZCGRP calls DFHZGRP
3. DFHZGRP issues VTAM INQUIRE
4. DFHZGRP either:

   terminates session via DFHZGUB issuing CLSDST/TERMSESS or

   restores the session with OPNDST TYPE=RESTORE
5. DFHZGRP queues restored sessions for further processing.
6. DFHZGRP issues RECEIVE_ANYs.
7. DFHZGRP does some CNOS work.

8. DFHZGRP does some URD work.
9. Queued sessions get restored.

More detail: Sessions that persist at startup time are processed by:

1. Attach task CGRP - program DFHZCGRP in DFHSII1 after TCRP is attached.
2. DFHZCGRP calls DFHZGRP with a START_TYPE of:

   COLD
   WARM
   EMER_XRF
   EMER
3. DFHZGRP issues VTAM INQUIREs in 'chunks', that is VTAM is passed an area whose size is defined in the TCT Prefix.

   The area is filled with NIBS by VTAM. DFHZGRP scans the NIBS and decides whether to UNBIND or OPNDST each session.

   For COLD, WARM and EMER_XRF all sessions are unbound.

   For EMER some sessions are unbound and some restored depending on the circumstances.
4. Restored sessions are queued to DFHZACT for further processing by DFHZXRC or DFHZXPS.
5. RECEIVE_ANY Initialization done.
6. CNOS records are processed by making calls to DFHZGPC.
7. URDS are reset to AWAITING RE_SYNCHRONIZATION for EMER only.
8. DFHZACT calls DFHZXRC or DFHZXPS for each session queued by DFHZGRP.

Task and module Flow diagram:

```
-> indicates an ATTACH

  TASK
                     TCRP
   1     TCP     III     CSSY          CGRP
         ---     ---     ----          ----
  .
  .
  SII1->ZCSTP
         ZDSP
         .ZSLS
         . ZGSL
         Spin on
         TCTV_RA_DONE
         .
  SII1---------->SII1 --->TCRP
                 SII1----------------->   ZCGRP
         .       .       . install       .ZGRP
         .       .       . TCTTEs        .  INQUIRE on
         .       .       . etc           .  persistent sessions
         .       .       .               .  wait on TCTVCECB (EMER)
         .       .       . Post TCTVCECB
         .       .       task end        .
         .       .                       .  process persistent
         .       .                       .  sessions
         .       .                       .  RECEIVE_ANY processing
         ZDSP continues <------------------- set TCTV_RA_DONE
                 .                          post TCTV_ZGRP_FIN_ECB
                 .                       task end
                 . Wait on
                 . TCTV_ZGRP_FIN_ECB
                 SIJ1
                 . SETLOGON START
                 . Start CXRE task
```

```
            . Control is Given to CICS
    ZACT
    . ZXRC
    . ZXPS
```

Task and module flow - more detail.:

1. Startup runs as normal until DFHSII1 has started the TCP (CSTP) task and DFHZDSP runs.
2. DFHZDSP calls DFHZSLS.
   - If VTAM is at least V3R4.1, DFHZSLS calls DFHZGSL to issue SETLOGON OPTCD=PERSIST if the SIT PSDINT value is a valid non 0 value.
   - If the VTAM level is V3R4.0 or PSDINT is 0 or defaulted with higher levels of VTAM, DFHZSLS calls DFHZGSL to issue SETLOGON OPTCD=NPERSIST.
   - If the VTAM level is lower than V3R4.0, the SETLOGON OPTCD call is not made since PERSIST and NPERSIST are not supported for these VTAM releases.

   DFHZSLS does NOT issue RECEIVE OPTCD=ANY. It returns to DFHZDSP which 'spins" until TCTV_RA_DONE is set by DFHZGRP when the RECEIVE_ANYs have been successfully issued.
3. DFHSII1 attaches the III task which continues to run code in DFHSII1.
4. DFHSII1 (III) attaches and calls DFHTCRP as a system task then attaches task CGRP, which runs program DFHZCGRP which calls ZGRP.
5. DFHZGRP calls DFHZGUB if there are any sessions to unbind.
6. DFHZGRP queues any sessions to be restored to DFHZACT.
7. DFHZGRP sets TCTV_RA_DONE after issuing RECEIVE_ANYs to allow DFHZDSP to continue.
8. DFHZGRP posts TCTV_ZGRP_FIN_ECB.
9. When DFHZGRP finishes, control is returned to code in DFHZCGRP.

   DFHZCGRP checks the RESPONSE and REASON code. It sets TCTV_ZGRP_FAILED off if RESPONSE(OK) or RESPONSE(EXCEPTION) with REASON(ACB_CLOSED|INQUIRE_FAILED). Otherwise it sets TCTV_ZGRP_FAILED on.
10. DFHSII1 waits on TCTV_ZGRP_FIN_ECB and checks TCTV_ZGRP_FAILED set by DFHSII1.

    If TCTV_ZGRP_FAILED is off then DFHSII1 continues Otherwise it sets INITDERR which causes CICS to terminate when the other tasks have finished.
11. Just before CONTROL IS GIVEN to CICS, DFHSIJ1 attaches the CXRE task to run DFHZXRE0 which does some additional PRSS processing.
12. DFHZXRC or DFHZXPS are then called to process any TCTTEs queued to DFHZACT.
13. DFHZXRC is called by DFHZACT to process non-APPC sessions which have not been unbound by DFHZGRP. It takes one of the following actions depending on the state of the session, the terminal type, and how the TYPETERM for the session has been defined to CICS.
    - Send END_BRACKET
    - Send CLEAR (followed by START_DATA_TRAFFIC for SNA devices which support it)
    - Unbind.

    For those devices for which the cleanup action is not to unbind, the TCTTE is queued to DFHZNAC and message DFHZC0146 is issued for the session.

    As part of the processing for message DFHZC0146, any recovery notification requested for the session is initiated:
    - If the requested recovery notification is MESSAGE, DFHZNCA sends a BMS map to the terminal.
    - If the requested recovery notification is TRANSACTION, DFHZNCA initiates the requested transaction.
14. DFHZXPS is called by DFHZACT to process APPC sessions.

    DFHZXPS takes one of the following courses of action depending on the setting of TCTE_PRSS on entry.

- Examine the data pointed to by TCTV_PRSS_CV29_PTR to determine the state of the session at system failure.
  a. If a task is attached call DFHZGDA to issue DEALLOCATE,ABEND for the task still running on the partner.
  b. If no task is attached but there is further recovery to be done e.g. bid recovery, outstanding responses, set the TCTTE to a state which allows this further recovery to proceed. If the existing mechanism will carry out the recovery without further intervention by DFHZXPS then remove the TCTTE from the DFHZACT queue, otherwise requeue the TCTTE to DFHZACT and DFHZXPS will be recalled at a later stage to finish recovery processing.
  c. If no task is attached and there is no further recovery to be done, remove the TCTTE from the DFHZACT queue as recovery is now complete.
- Recall DFHZGDA to continue with DEALLOCATE,ABEND or REJECT_ATTACH processing.
- Requeue the TCTTE to DFHZACT if a SEND (for example of an outstanding response) which was set in motion by an earlier instance of DFHZXPS is still in progress.
- CLSDST the session if an error has occurred during the recovery process.
- Carry out further recovery as described above, if required, following successful completion of DEALLOCATE,ABEND processing.
- Remove the TCTTE from the DFHZACT queue when all recovery has completed.

Sessions that persist at Dynamic Open

If VTAM fails but CICS stays up SNPS sessions do not exist. For MNPS they do persist. When VTAM crashes, CICS does not delete the autoinstalled resources and resets all the terminal and connection sessions to unopened state.

Summary:

1. CEMT SET VTAM OPEN
2. DFHEIQVT calls DFHZOPA
3. DFHZOPA calls DFHZSLS
4. DFHZSLS call DFHZGSL
5. DFHZGSL issues SETLOGON PERSIST or NPERSIST
6. DFHZOPA calls DFHZGRP
7. DFHZGRP issues INQUIRE PERSESS
8. DFHZGRP terminates session via DFHZGUB issuing CLSDST/TERMSESS. However, if MNPS is in use the sessions are OPNDST RESTOREd instead.
9. DFHZGRP issues RECEIVE_ANYs
10. DFHZGRP deletes CNOS catalogue records
11. DFHZOPA issues SETLOGON START

More detail: Sessions that persist after the ACB has been opened using CEMT SET VTAM OPEN or EXEC CICS SET VTAM OPEN are processed by:

1. CICS is running with the VTAM ACB closed.

   CEMT SET VTAM OPEN or EXEC CICS SET VTAM OPEN is issued.
2. DFHEIQVT calls DFHZOPA to open the ACB.
3. DFHZOPA calls DFHZSLS.
4. DFHZSLS calls DFHZGSL.
5. DFHZGSL issues VTAM macro calls dependent upon the VTAM level and PSDINT value.
   - If VTAM is at least V3R4.1, DFHZGSL issues SETLOGON OPTCD=PERSIST if the SIT PSDINT value is a valid non 0 value.

- If the VTAM level is V3R4.0 or PSDINT is 0 or defaulted with higher levels of VTAM, DFHZGSL issues SETLOGON OPTCD=NPERSIST.
- If the VTAM level is lower than V3R4.0, the SETLOGON OPTCD call is not made since PERSIST and NPERSIST are not supported for these VTAM releases.

6. DFHZOPA then calls DFHZGRP with startup type of DYNOPEN.
7. DFHZGRP issues INQUIRE PERSESS with a storage area that will take up to about 400 sessions - INQUIRE PERSESS is reissued until all the NIBs have been obtained from VTAM.
8. DFHZGRP calls DFHZGUB if there are any sessions to unbind. For MNPS DFHZGRP instead issues OPNDST RESTORE for each session that persists.
9. DFHZGRP issues RECEIVE_ANYs.
10. DFHZGRP calls DFHZGCC to delete CNOS records.
11. If ZGRP returns RESPONSE(OK) or RESPONSE(EXCEPTION) with REASON(ACB_CLOSED|INQUIRE_FAILED) then DFHZOPA issues SETLOGON OPTCD=START. Otherwise it causes DFHZSHU to be run to close the VTAM ACB and then returns to DFHEIQVT.

TCB Concurrency

Summary: If SUBTSKS = 1 Specified in SIT

- DFHZGRP switches to concurrent TCB if enough NIBS to process.
- INQUIRE PERSESS work done concurrently with TCRP ZC INSTALL.
- DFHZGUB switches to concurrent TCB if enough NIBS to process. (EMER only).
- OPNDST RESTORE and CLSDST/TERMSESS done concurrently.

More detail: During startup DFHZGRP is attached as a task and runs at the same time as other startup tasks such as DFHTCRP and DFHRCRP. However, DFHZGRP also switches to use the CONCURRENT TCB if there are enough NIBS to process during EMER start.

This allows DFHZGRP to issue INQUIRE OPTCD=PERSESS as many times as is necessary, concurrently with the TCTTEs being restored by DFHTCRP.

When DFHZGRP finishes INQUIREing it waits for DFHTCRP to finish before matching each persisting NIB with the restored TCTTEs.

Each NIBLIST is then OPNDST OPTCD=RESTOREd and while this is running asynchronously DFHZGUB is called to run under the concurrent TCB if there are enough NIBs to be unbound in the NIBLIST.

Persistent Signon under Persistent Sessions

1. After the persistent session has been recovered, the TCTTE is marked to indicate that the signon will persist.
2. The RECOVNOTIFY message or transaction is processed.

   Note: Because RECOVNOTIFY is processed before persistent signon is recovered, only the first transaction specified in the RMTRAN system initialization parameter will be processed; the second transaction specified cannot be processed because security has not yet been restored yet.
3. The user presses an Attention IDentifier (AID) key.
4. CICS runs the CPSS transaction to recover the signon.

Modules

ZC (terminal control) together with the following:

Module	Function
DFHZCGRP	Program initiated by task CGRP to set up the start type and to call DFHZGRP during initialization. It then analysis the response from DFHZGRP and decides if CICS can continue or not.
DFHZGCA	Sets the appropriate ZC control blocks to reflect the currently agreed Change Number Of Session (CNOS) values for an LU6.2 connection.
DFHZGCC	Performs catalogue and retrieval of CNOS data. This module is called when CICS needs either to store or to recover CNOS values. During a CICS run, all CNOS values are written to the global catalogue. Under normal circumstances they are not needed. However, if a persistent sessions restart is performed, it is necessary to recover the CNOS values which were in operation at the time of the CICS failure. This is achieved by having a record on the global catalogue which can be read in during PRSS restart and used to restore the sessions to their pre-failure state. This module will handle the maintenance of the CNOS records during normal CICS operation and the recovery of the records during PRSS recovery.
DFHZGCN	Handles the process of LU6.2 Change Number Of Sessions (CNOS) negotiation, acting as either the source or target end of the conversation, and calls DFHZGCA to action the resulting changes.
DFHZGDA	The role of DFHZGDA is to take control of APPC conversations which have persisted across a CICS failure, and to ensure that they are terminated cleanly, by issuing a Deallocate(Abend) informing the partner LU that the CICS transaction has abended. If DFHZGDA is working correctly, the fact that CICS has failed and been restarted should be transparent to the partner LU; all he knows is that the CICS transaction to which he was talking has terminated. DFHZGDA also performs REJECT_ATTACH processing for synclevel 2 conversation which are started by the partner before Exchange Lognames has been done after a persistent sessions restart.
DFHZGPC	Performs recovery of CNOS values for modegroups. This module is called when CICS is performing a persistent sessions (PRSS) restart. When a PRSS restart is performed, it is necessary to do more than merely recover the session. It is also necessary to recover the CNOS state which the sessions had prior to the CICS failure. DFHZGCC will have maintained a record of the CNOS state on the global catalogue. This record will now be used in this module in an attempt to restore CNOS values.
DFHZGPR	The role of DFHZGPR is to update the global catalog whenever it is necessary to add, delete, or test for a record indicating that an APPC connection has a Persistent Resource associated with it. A Persistent Resource can be defined as some session state, or piece of work upon which the partner LU is dependent, and which will be lost in the event of CICS failing. The only Persistent Resource so far identified is: • A shipped AID Prior to persistent sessions, the failure of the APPC session tells the partner that these resources have been lost, and drives his recovery. With the advent of persistent sessions, it is necessary for a persisting CICS to know that an APPC session had a Persistent Resource associated with it, so that the connection can be unbound (to drive the partners cleanup) and then rebound.

VTAM persistent sessions

Module	Function
DFHZGRP	Initialize VTAM persistent sessions. DFHZGRP is a domain subroutine but is called by DFHZCGRP (task CGRP) during initialization. DFHZGRP is called during ZC initialization or when the VTAM ACB is opened dynamically by CEMT SET VTAM OPEN or EXEC CICS SET VTAM OPEN by DFHEIQVT. The module does the following: 1. OPNDST RESTOREs or CLSDST/TERMSESS any session that VTAM has held persisting, depending on start up type and session parameters. 2. It calls DFHZGPC to re-instate CNOS records during an EMER restart, or calls DFHZGCC to delete CNOS catalogue records. 3. It initializes the RECEIVE_ANY RPLs and issues the RECEIVE_ANYs.
DFHZGSL	Informs VTAM whether sessions are to persist or not. This module is called when CICS needs to set, unset or change the Persistent Sessions PSTIMER value.
DFHZGUB	Issue CLSDST or TERMSESS for individual NIBs in a NIBLIST. This module is called by DFHZGRP to unbind nibs in a niblist in two ways: • Unbind the entire NIBLIST for COLD, WARM, EMER+XRF and dynamic open. • Unbind only the NIBs with NIBUSER = 0 for EMER starts.
DFHZXPS	DFHZXPS handles Persistent Sessions recovery for APPC sessions. It does not deal with non-APPC sessions which are dealt with by DFHZXRC. DFHZXPS is called by DFHZACT after OPNDST OPTCD=RESTORE has been issued successfully for a persisting APPC session. Both single and parallel APPC sessions are dealt with but there is no difference in the processing. The task of DFHZXPS is to examine VTAM session tracking data which was hung off TCTE_PRSS_CV29_PTR by DFHZGRP following a Persistent Sessions restart and if possible to update the TCTTE to allow work to continue on the session. If it is not possible to determine the state of the session prior to system failure, or the session was not in a state which allows it to be recovered, the session will be unbound.
DFHZXRC	DFHZXRC analyses the Session State Vector data that is hung off TCTE_PRSS_CV29_PTR by DFHZGRP during an EMER restart, for each persisting session. The necessary action to cleanup and recover the session is then initiated.

Diagnosing Persistent Sessions Problems

The following should be consulted when diagnosing problems with persistent session.

- Trace, TC level 1, 2 and exception in the range of AP FB10-FBFF.
- CEMT INQUIRE VTAM showing the PSDINT value.
- Console and CSNE logs:
  - Persistent session messages (DFHZC0001 to DFHZC0162)
  - Information produced by DFHZNAC
- Dumps taken by some of the above messages.

  If a NIBLIST was present at the time the dump was taken then it can be examined by printing the TCP section of the dump.
- Last flow information - that is the CV29, FMH5, BIS and BID information is useful if a session is in the wrong state after a persistent session restart. This may have been diagnosed by an error message, or maybe missed and message DFHZC0146 or DFHZC0156 issued.

TCTE_PRSS_CV29_PTR points to the CV29 etc which was created by DFHZGRP and used by DFHZXPS or DFHZXRC. It is freed when DFHZNAC issues message DFHZC0146 or DFHZC0156. Otherwise it is freed when the session is unbound.

It is traced by DFHZXPS as a TC level 1 trace.

If you have a dump, but no trace level 1 available, it is dumped in the TCP section for each TCTTE for which it still exists.

- The contents of byte TCTE_PRSS are useful. Values other than X'00' and X'FF' indicate that something went wrong during the PRSS recovery. The possible values are listed in the *CICS Data Areas*. If a value is left in this byte, the meaning may give some indication as to where the recovery went wrong. The values are described later in this chapter.
- The contents of the state machines are useful.

  TCTECNTS - contention state machine.

  TCTEBKTS - bracket state machine.

  TCTECHSS - chain state machine.

  TCTEUSRS - user state machine.
- The contents of TCTE_BID_STATUS are useful. They are described later in this chapter.

Here are some possible problems:

- DFHZGRP may cause CICS to terminate during initialization for the following reasons:
  - DFHZGRP has been called with invalid parameters.
  - DFHZGRP is unable to complete the receive any process.
  - DFHZGRP has had a loop or abend.
  - DFHZGRP is unable to switch back to the QR TCB.
  - DFHZGRP has failed before any NIBs have been obtained from VTAM (with INQUIRE OPTCD=PERSESS).
  - DFHZGRP or DFHZGUB has issued a VTAM request that failed to respond within 5 minutes. Issued with message DFHZC0128 and a system dump.

  In each case DFHZGRP or a function it has called issues a message giving a reason for the failure.
- Sessions may be unbound by DFHZGRP for the following reasons:
  - This is a COLD, WARM, EMER + XRF restart.
  - This is a dynamic open of the ACB (e.g. CEMT SET VTAM OPE). However, if MNPS is in use sessions should be restored at this point.
  - The TCTTE has not been found - probably because it has not been cataloged (Autoinstall with AIRDELAY=0 or APPC clone). No message is written because this is considered to be normal.
  - CICS does not support recovery for LU61 or pipeline sessions.
  - The TCTTE does not match the NIB - possibly an operational mix-up - has the right GCD been used?
  - A terminal or session had RECOVOPT UNCONDREL|NONE specified.
  - A connection had PSRECOVERY NONE specified.
  - A matching mode group was not found - have you got the right GCD?
  - A suitable session was not found - this can occur if the CNOS values create many “up for grabs” sessions which were in use when CICS failed - this would occur if the session limit was high and the contention winners was low.

    It may also occur if CICS was in the process of CNOSing from a high session limit to a low session limit at the time CICS failed.

    Message DFHZC0111 is issued in this case.
  - An URD was found for the session so the entire connection is unbound to allow the connection to recover correctly.
- APPC Sessions may be unbound by DFHZXPS for the following reasons:

Some of the reasons are known states for which the session cannot be recovered. Others are unexpected errors.

Known states for which the session cannot be recovered:.
- The last flow was a positive response to a bid with data.
- Exchange log names (transaction CLS2) was running when the system failed.
- A bind or bind security had not completed when the system failed.
- Because of the last thing to flow e.g. SIGNAL, the state of the session at the time of system failure cannot be determined.

Unexpected errors:
- A bad return code was received from a call to DFGZGDA.
- An attempt to reset the session from CS mode to CA mode or vice versa failed.
- The TCTE_PRSS byte contained an unexpected value on entry to DFHZXPS.
- The BIS, bid or CV29 data pointed to by TCTV_PRSS_CV29_PTR contained an unknown value or was inconsistent.
- An error occurred during some other part of the recovery process.
- An internal logic error occurred in DFHZXPS.

- Sessions may be unbound by DFHZGDA for the following reasons:
  - A SEND issued as part of Deallocate(Abend) processing has failed
  - A RECEIVE issued as part of Deallocate(Abend) processing has failed
  - A logic error is detected during Deallocate(Abend) processing
- Sessions may be unbound by DFHZXRC for the following reasons:
  - The user has specified RECOVOPT(RELEASESESS) and the session was in bracket at the time CICS failed.
  - End-Bracket and Clear/SDT could not be used to clean up the session.
  - Cold Start has been requested for the session.
- Message DFHZC0124 can be issued with inconsistent counts if:
  - DFHZGRP loops or abends.
  - The ACB is closed by VTAM operator commands whilst DFHZGRP is in control.
- LU6.2 Connections which might be expected to persist, may be unbound if there was a persistent resource associated with the connection when CICS failed (i.e. there was an asynchronous processing request in progress at the time CICS failed).
- Following a persistent sessions restart, LU6.2 partners may experience a series of unexpected abends with sense code 08640001 from the persisting CICS; this can occur either because there was a conversation in progress at the time CICS failed, and CICS has terminated the conversation with this code, or for synclevel 2 conversations, the partner has attempted to initiate a conversation before Exchange Lognames has run following a persistent sessions restart.
- Some APPC sessions may hang following a persistent sessions restart because CICS has determined that it was in RECEIVE state at the time of the CICS failure, and issued a RECEIVE for the expected data, but the partner has not sent the expected data; the RECEIVE will not timeout in this situation, because RTIMOUT does not apply to sends issued by DFHZGDA.

Persistent Sessions status byte (TCTE_PRSS)

A new byte TCTE_PRSS has been introduced into the TCTTE to track the stage reached in the persistent sessions recovery of a session. If for some reason persistent sessions recovery does not complete, this field can give a useful indication of the stage reached in recovery when the problem occurred.

TCTE_NO_PRSS_RECOVERY (X'00')
: The value TCTE_PRSS would normally contain, meaning:
  - Persistent sessions are not being used.

- The session was successfully recovered following a persistent sessions restart.
- The session has been CLSDSTed and restarted since a persistent sessions restart.
- The session was started after any persistent sessions restart.

If this was a persisting VTAM session, then TCTE_PRSS will have been set to this value on completion of recovery notification for non-LU6.2 (see NAPES84 and NAPES83 routines), or in the session restarted logic of NAPES51 for LU6.2 sessions.

TCTE_NIB_MATCHED (X'01')
Placed in TCTE_PRSS by DFHZGRP once a TCTTE has been found which matches the NIB of a persisting VTAM session. This should be a transient value, as the OPNDST OPTCD=RESTORE is issued soon after, and should cause TCTE_PRSS to be updated.

TCTE_OPNDST_RESTORE_COMPLETED (X'02')
Placed in TCTE_PRSS once an OPNDST OPTCD=RESTORE has been successfully issued for a VTAM Session by DFHZGRP. Once this value has been placed in TCTE_PRSS, the TCTTE should be put onto the activate scan queue to await processing by DFHZXRC or DFHZXPS.

TCTE_ZXRC_CLEANUP (X'20')
Placed in TCTE_PRSS by DFHZXRC when it begins processing a TCTTE. All TCTE_PRSS values relating to DFHZXRC processing are X'2x'. This value remains in TCTE_PRSS until the TCTTE is queued to DFHZNAC for the issuing of message DFHZC0146. If for some reason the TCTTE does not get recovered and TCTE_PRSS contains this value, then DFHZXRC may be the culprit.

TCTE_ZXRC_ISSUE_RECOVERY_MSG (X'21')
DFHZXRC has identified the cleanup and recovery actions required, and has queued the TCTTE to DFHZNAC for recovery message processing (message DFHZC0146). If there is any problem with the recovery notification processing in DFHZNCA, then TCTE_PRSS is likely to contain this value; it may be that the TCTTE has been taken off the DFHZACT or DFHZNAC queues for an unexpected reason.

TCTE_ZXPS_CLEANUP (X'30')
All TCTE_PRSS values beginning (X'3x') indicate that DFHZXPS is doing its recovery/cleanup processing for this TCTTE. TCTE_PRSS is updated to this value on entering DFHZXPS for the first time. DFHZXPS should only be processing LU6.2 sessions.

TCTE_ZXPS_DEALLOCATE_ABEND (X'31')
DFHZXPS places this value into TCTE_PRSS prior to calling DFHZGDA for the first time. It indicates that DFHZXPS has determined that an APPC conversation was taking place at the time CICS failed, and that DFHZXPS is calling DFHZGDA to terminate that conversation. Again, this should be a transient value, as DFHZGDA will update TCTE_PRSS as it proceeds with its DEALLOCATE(ABEND) processing.

TCTE_ZXPS_SEND_IN_PROGRESS (X'32')
DFHZXPS has determined that bidding activity was taking place at the time CICS failed, and that some kind of SEND is required to complete the bid flows. If the session hangs with this value in TCTE_PRSS there may have been some kind of problem with unexpected bid flows taking place.

TCTE_ZXPS_ISSUE_RECOVERY_MSG (X'33')
When DFHZXPS has completed recovery and cleanup for the session, it puts this value into TCTE_PRSS before queueing the TCTTE to DFHZNAC for recovery message processing.

TCTE_ZGDA_FMH7_SEND (X'41')
All TCTE_PRSS values with X'4x' indicate that DFHZGDA is terminating the APPC conversation which was in progress on the session at the time CICS failed. This value indicates that DFHZGDA is in the process of issuing a SEND for the FMH7 which is to terminate the conversation.

TCTE_ZGDA_FMH7_COMP (X'42')
DFHZGDA has completed its Deallocate(Abend) processing. This value in TCTE_PRSS indicates to DFHZXPS that it may continue with any outstanding recovery/cleanup processing of its own.

TCTE_ZGA_FMH7_REC (X'43')
DFHZGDA has determined that CICS was in RECEIVE state at the time CICS failed, and has issued a RECEIVE for the RU expected from the partner. This value may appear in sessions which appear to be hanging following a persistent sessions restart. If the partner never issues the expected SEND, the RECEIVE will never be executed. Since this RECEIVE is issued under the TCP task, the RECEIVE will not be subject to any RTIMEOUT.

TCTE_ZGDA_REC_EOC (X'44')
Placed in TCTE_PRSS if the first RECEIVE of the DFHZGDA module following the persistent sessions reveals that the partner is in the middle of sending a chain of RUs. If TCTE_PRSS contains this value, DFHZGDA has issued a RECEIVE_PURGE for the session. Again, depending on how quickly the partner sends the expected data, this session may appear to hang.

TCTE_ZGDA_SEND_RESP (X'45')
Placed in TCTE_PRSS if DFHZGDA has to issue a SEND for a response during Deallocate(Abend) processing.

TCTE_PRSS_CLSDST_SCHEDULED (X'FF')
This value is placed in TCTE_PRSS if there is an error, or if in the course of persistent sessions recovery it is decided to terminate the persisting session. This may be for a variety of reasons; some of which are:

- An error occurred issuing a SEND or RECEIVE during persistent sessions recovery.
- RECOVOPT(NONE) or RECOVOPT(UNCONDREL) was specified for the session.
- The only recovery action which DFHZXRC could take was to terminate the session.

The X'FF' value remains in TCTE_PRSS as an indicator that the session was terminated during PRSS recovery. Only when the session is restarted is the value overwritten with X'00'.

Bid status byte (TCTE_BID_STATUS)

DFHZXPS uses a byte in the TCTTE, TCTE_BID_STATUS, to track the various stages of recovery. It is possible to examine this byte to determine the stage of recovery reached by DFHZXPS.

The byte values have the following meanings.

- X'00'

  This session has not been processed by DFHZXPS.
- X'01' TCTE_SEND_POSITIVE_RESPONSE

  A positive response is to be sent to a bid which was received before system failure. This value is changed to X'07' TCTE_SENT_POSITIVE_RESPONSE before the TCTTE is requeued to DFHZACT for the SEND and so will only be seen if DFHZXPS abends. When the response has been sent DFHZXPS will be recalled.
- X'02' TCTE_SEND_NEGATIVE_RESPONSE

  A negative response is to be sent to a bid with data which was sent before system failure. This needs to be followed by RTR and so the status byte is changed to X'03' SEND_RTR before the TCTTE is requeued to DFHZACT for the SEND. This is another value which should only be seen if DFHZXPS abends. DFHZXPS will be recalled when the response has been sent.
- X'03' TCTE_SEND_RTR

  Recovery is complete apart from the need to send RTR. This will be done by DFHZDET and DFHZXPS will not be recalled.
- X'04' TCTE_SENT_RTR

  RTR was sent before system failure. There is no recovery to be done. DFHZXPS will not be recalled.
- X'05' TCTE_SEND_LUSTAT_EB

Either we received a positive response to a bid or we sent a positive response to RTR before the system failed. The bid now has to be canceled. This will be done by DFHZDET and DFHZXPS will not be recalled.

- X'06' TCTE_AWAITING_BB_RESPONSE

  A bid was sent before the system failed. No further recovery is required. When the response arrives from the partner the bid will be canceled. DFHZXPS will not be recalled.
- X'07' TCTE_SENT_POSITIVE_RESPONSE

  Either a positive response has been sent to a bid or one is about to be sent (see above under SEND_POSITIVE_RESPONSE). In the former case DFHZXPS will not be recalled, in the latter case it will.
- X'08' TCTE_0814_RECEIVED

  A negative response was sent to a bid before the system failed. Any further recovery will be carried out by DFHZDET and DFHZXPS will not be recalled.
- X'09' TCTE_0813_RECEIVED

  As above except that no RTR is expected in this case. No further recovery processing is needed from either DFHZXPS or DFHZDET.
- X'0A' TCTE_SEND_RECOVERY_MESSAGE

  All recovery is now complete.
- X'0B' TCTE_DR1_OUTSTANDING

  The last flow was inbound with CEB,RQD1 and so although there is no task to ABEND a response is still expected by the partner. We requeue for DFHZSDL to send the response and any further recovery processing will be done by DFHZDET. DFHZXPS will not be recalled.
- X'0C' TCTE_DR1_EXPECTED

  As above except that the last flow was inbound. DFHZDET will arrange for the response to be received. DFHZXPS will not be recalled.

TCTE_BID_STATUS must be used in conjunction with TCTE_PRSS to determine the state of the recovery. If TCTE_PRSS is set to TCTE_ZXPS_ISSUE_RECOVERY_MESSAGE, or to a state which indicates that recovery is complete, DFHZXPS has finished processing. If not DFHZXPS will be recalled at a later stage.

Summary of persistent session waits

The DFHDSSRM waits are summarized here. All but PSUNBECB are posted by DFHZGRP.

```
Module  Type Resource_name Resource_type ECB

DFHSII1 MVS  ZGRPECB       AP_INIT       TCTV_ZGRP_FIN_ECB

DFHZGUB OLDC PSUNBECB      ZC_ZGUB       WAIT_RPL_ECB

DFHZGRP MVS  PSOP1ECB      ZC_ZGRP       OPNDST_ECB

DFHZGRP MVS  PSOP2ECB      ZC_ZGRP       OPNDST_ECB

DFHZGRP MVS  PSINQECB      ZC_ZGRP       INQUIRE_ECB

DFHZGRP OLDC TCTVCECB      ZC_ZGRP       TCTVCECB
```

where the waits are issued for the following reasons:

ZGRPECB Wait for DFHZGRP to complete.

PSUNBECB Wait for free unbind RPL from RPL pool anchored from TCTV_PRSS_RPL_POOL_PTR.

PSOP1ECB Wait for OPNDST RESTORE to complete.

PSOP2ECB Wait for OPNDST RESTORE to complete after UNBINDs have failed.

PSINQECB Wait for INQUIRE PERSESS to complete.

TCTVCECB Wait for TCTTEs to finish installing (DFHTCRP).

VTAM exits

The VTAM exits SYNAD (DFHZSYX) or LERAD (DFHZLEX) may be driven during persistent sessions recovery.

In DFHZGRP, before INQUIRE OPTCD=PERSIST is issued, or in DFHZGUB before CLSDST or TERMSESS are issued CICS sets the RPL user field to -2 to indicate to the exits that they must do NO processing at all. This is because these macros may be issued under the concurrent TCB.

In DFHZGRP before OPNDST OPTCD=RESTORE is issued CICS sets the RPL user field to -1 to indicate to the exits that they should try minimum recovery - that is they set the return code to TCZSYXPR if an error can be retried, or TCZSYXCF if it is a permanent error.

If an error occurs in DFHZGSL for SETLOGON OPTCD=PERSIST DFHZSYX returns immediately (as for RPL user field = -2).

If MNPS is in use and VTAM crashes DFHZTPX is driven with a code of 8. If SIT parameter PSTYPE=MNPS was specificed then DFHZTPX does NOT schedule the autoinstalled TCTTEs for deletion. They are scheduled for CLSDST CLEANUP instead by DFHZSHU.

See the *OS/390 eNetwork Communications Server: SNA Programming* manual, SC31-8573, for general VTAM exit information.

Trace

The following point IDs are provided for persistent sessions recovery (DFHZGCA, DFHZGCC, DFHZGCN, DFHZGDA, DFHZGPC, DFHZGPR, DFHZGRP, DFHZGSL, DFHZGUB, DFHZCGRP, DFHZXPS, DFHZXRC):

- AP FB10 through AP FBFF, for which the trace levels are TC 1 and TC 2.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Statistics

The following statistics are produced by DFHZGRP. They are treated in the same way as other terminal control VTAM statistics.

A03_PRSS_NIB_COUNT The number of active VTAM sessions when INQUIRE OPTCD=COUNTS was issued - this represents the number of persisting sessions.

A03_PRSS_INQUIRE_COUNT
The number of times DFHZGRP issues INQURE OPTCD=PERSESS. Each INQUIRE should be given about 400 sessions.

A03_PRSS_UNBIND_COUNT The number of times CLSDST or TERMSESS were issued by DFHZGUB.

A03_PRSS_OPNDST_COUNT
The number of sessions that OPNDST RESTORE restored successfully.

A03_PRSS_ERROR_COUNT The number of sessions with NIBUSER=tctte address, that VTAM failed to restore with OPNDST RESTORE. This occurs if VTAM operator commands are issued whilst DFHZGRP is in control and sessions are closed as a result.

Chapter 68. WTO and WTOR

Design overview

The DFHSUWT module provides the following support for executing MVS WTO and WTOR SVCs:

SEND
supports Write To Operator (WTO):
- A single-line message up to 113 characters, or a multiline message consisting of a control line and up to nine lines of 69 characters
- Route code specification (route code list of 1 through 28 numbers, each in the range 1 through 28)
- Descriptor code specification (descriptor code list of 1 through 16 numbers, each in the range 1 through 16).

CONVERSE
supports Write To Operator With Reply (WTOR):
- A single-line message up to 121 characters
- Route code specification (route code list of 1 through 28 numbers, each in the range 1 through 28)
- Descriptor code specification (descriptor code list of 1 through each in the range 1 through 28) 16 numbers, each in the range 1 through 16)
- A reply with maximum length of 119 characters.

The DFHWTO macro may be used to send a message, normally to the system operator, when neither the CICS message domain nor the old message program (DFHMGP) can be used. The message domain cannot be used during certain phases of initialization and XRF processing, because it requires a kernel stack environment. DFHMGP cannot be used during initialization, nor during any sort of abend or dump processing, because it uses task LIFO storage and may therefore invoke the storage control program.

The DFHWTO macro may also be used to terminate CICS abnormally or to request a reply from the operator.

Any WTO or WTOR macros that are issued by CICS might be intercepted by the console message handling facility described under “Console message handling” on page 359. This service optionally inserts the CICS region’s applid into CICS messages before they are displayed on the console.

Modules

DFHSUWT and DFHWTO

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:
- AP FF0x, for which the trace levels are AP 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 69. CICS Web support and the CICS business logic interface

CICS Web support is a collection of CICS services that enable a CICS region to act both as an HTTP server, and as an HTTP client. When CICS is an HTTP server, Web clients can use transaction processing services by calling CICS programs or by running CICS transactions. When CICS is an HTTP client, a user application program in CICS can initiate a request to an HTTP server, and receive a response from it. Web clients use TCP/IP to communicate with CICS Web support.

The CICS business logic interface allows other external users to use transaction processing services.

Design overview

For information about the design and implementation of CICS Web support and the CICS business logic interface, see the *CICS Internet Guide*.

Control blocks

Figure 106 on page 506 shows the control blocks used by CICS Web support for 3270 display applications.

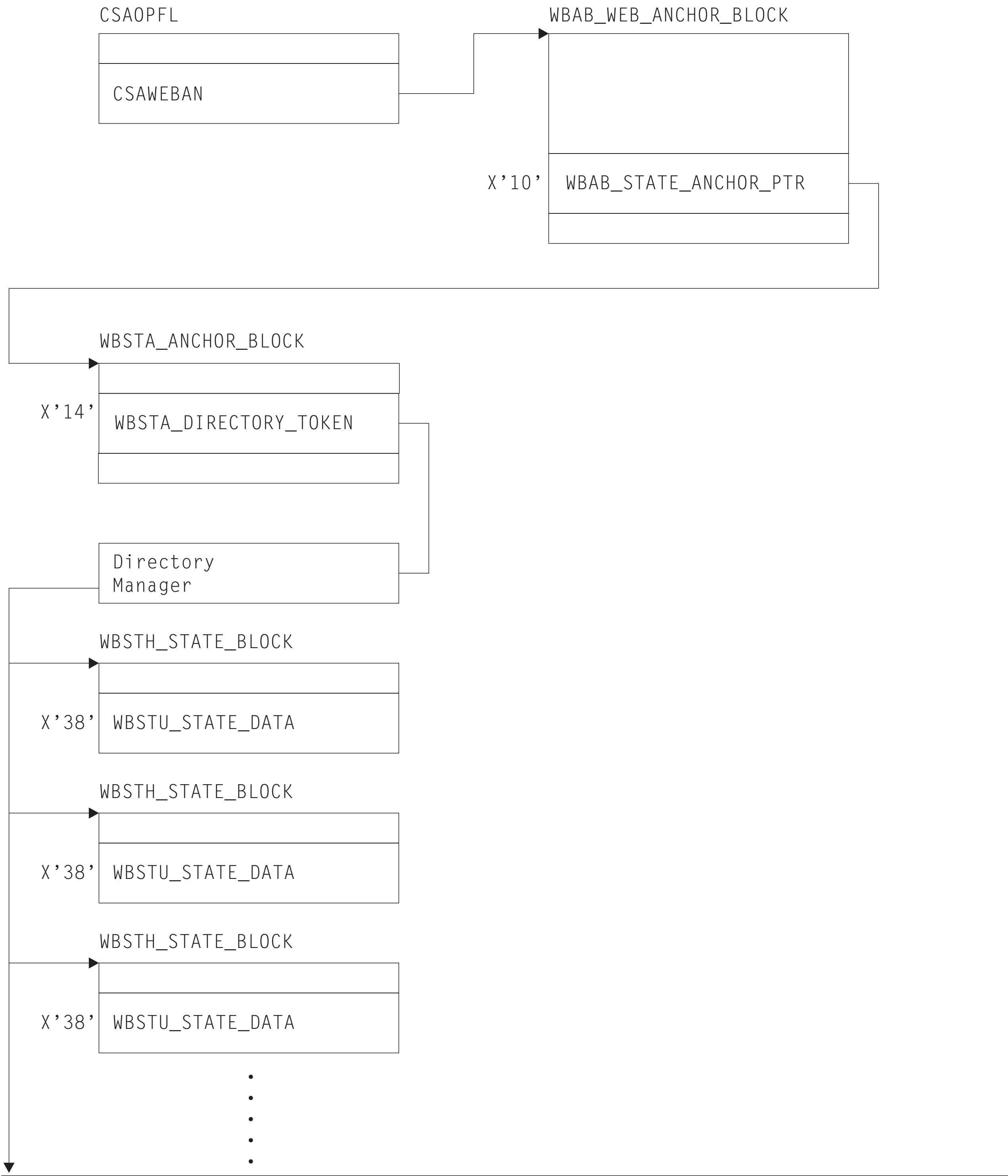


Figure 106. Web support module list

Modules

CICS Web support includes modules used for:
1. Initialization
2. Web attach processing
3. Default analyzer program
4. Alias transaction
5. Web error program
6. HTTP client processing
7. CICS business logic interface
8. CICS Web 3270 support
9. Unescaping function

Initialization, DFHWBIP

DFHWBIP initializes the Web environment at CICS startup.

Web attach processing, DFHWBXN

DFHWBXN is the Web attach processing module. It is the initial program invoked for transaction CWXN (or an alias of CWXN), which is attached for a new sockets connection received on a port associated with a TCPIPSERVICE definition with PROTOCOL(HTTP). It is also invoked for transaction CWXU (or alias), which is attached when the TCPIPSERVICE definition specifies PROTOCOL(USER). It calls the Web domain WBSR gate to process the incoming data.

Default analyzer program, DFHWBAAX

DFHWBAAX is the default analyzer program for a TCPIPSERVICE definition that specifies PROTOCOL(HTTP). It does not carry out further processing when a matching URIMAP definition has been found for the request, even if the URIMAP specifies ANALYZER(YES). It tests for the presence of a URIMAP definition, and if the result is positive, returns without performing any analysis on the request URL. This means that the settings specified in the URIMAP definition for the alias transaction, converter program and application program are automatically accepted and used to determine subsequent processing stages.

If no matching URIMAP definition is found, DFHWBAAX gives control to the user-replaceable Web error application program DFHWBERX to produce an error response. This is achieved by setting DFHWBERX as the application program to handle the request.

An alternative analyzer program that has been specified on the TCPIPSERVICE definition, such as the CICS-supplied sample analyzer program DFHWBADX, might carry out analysis on the request and specify alternative settings for the alias transaction, converter program and application program.

When the TCPIPSERVICE definition specifies PROTOCOL(USER), an analyzer program is always required to determine processing for requests (which are treated as non-HTTP requests). DFHWBAAX is not suitable for PROTOCOL(USER). The CICS-supplied sample analyzer program DFHWBADX or a customized analyzer program must be used instead. URIMAP definitions are not used with PROTOCOL(USER).

Alias transaction, DFHWBA

DFHWBA is the alias program. An alias transaction is started by Web attach processing for each request received from TCP/IP. The transaction ID can be selected by a URIMAP definition or an analyzer program, and the default is CWBA. For CICS Web support, DFHWBA calls the user application program that is specified to process the request. This application program could be specified in a URIMAP definition, or by an analyzer program or converter program. For the CICS business logic interface, DFHWBA calls the CICS business logic interface program.

Web error program, DFHWBEP

A Web error program is used to provide an error response to the Web client when a request error or an abend occurs in the CICS Web support process. DFHWBEP is called when CICS detects an error in request processing. Alternatively, in situations where the error can be identified by the active program, the Web error application program DFHWBERX can be specified to send the error response. DFHWBERX is used when the CICS-supplied default analyzer DFHWBAAX is specified as the analyzer program on the TCPIPSERVICE definition, and no matching URIMAP definition is found for a request.

DFHWBEP is called in the following situations:

- When CICS Web support detects a problem in initial processing of a request from a Web client, for example, if required information is missing from the request, or if the receive timeout is reached. This initial receive processing takes place before any utility programs (analyzer, converter or application programs) are called, and before URIMAP matching takes place.
- When an installed URIMAP definition matches the request, but the URIMAP definition or virtual host is disabled, or the resource for a static response is disabled or not found. No utility programs have been called at this stage.
- When neither the URIMAP definition, nor the analyzer and converter program processing, manages to determine what application program should be executed to service the request.
- When an abend occurs in the analyzer program, converter program, or user-written application program. This is indicated by a non-OK return code from the analyzer or converter program, or if these are not involved in request processing, by the failure of the application program to produce an acceptable HTTP response. Return codes set by analyzer or converter programs map to default HTTP responses that are built by CICS and passed through DFHWBEP for modification if necessary.

If a sockets send or receive error occurs, the socket is closed and no response is sent to the Web client.

DFHWBEP is user-replaceable. It uses a COMMAREA-based interface, where information about the error situation is passed to the program in a COMMAREA, and a complete HTTP response is created as a buffer of data. (DFHWBERX uses the EXEC CICS commands to obtain information about the Web client's request and create and send the error response.)

After the error response has been sent, the socket is usually left open for possible further requests from the Web client, until timeout is reached. The exception is when a 501 (Method Not Implemented) response is sent, in which case the socket is closed.

HTTP client processing, DFHWBCL

DFHWBCL is the HTTP client processing module. It is called by the command interface DFHEIWB (when EXEC CICS WEB commands with the SESSTOKEN option are used in application programs), and the COMMAREA interface DFHWBCLI, to handle outbound HTTP functions, such as opening a session and writing a request to the socket.

CICS business logic interface, DFHWBBLI

DFHWBBLI is the CICS business logic interface program. The interface to the CICS business logic interface program is described in *CICS Internet Guide*.

The CICS business logic interface program is called by DFHWBA. It calls the **Decode** function of a converter program, a CICS application program, or the **Encode** function of a converter program, according to what is specified in its parameter list, and passes the data back to the caller.

DFHWBA1 is the business logic compatibility interface program. In earlier releases, it was the business logic interface program, but it is now a compatibility layer on DFHWBBLI. It accepts data from an old-format business logic interface parameter list, copies it to the new format parameter list, then links to DFHWBBLI.

CICS Web support for 3270 display applications

The modules used by CICS Web support for handling 3270 display applications (sometimes referred to as the CICS Web bridge) are:

DFHWBGB
Removes redundant state data from the system.

DFHWBST
Manages the state data.

DFHWBTC
Performs conversion between 3270 and HTML.

DFHWBTTA
The Web terminal translation application, which sets up the parameters for bridging to transactions from CICS Web support. The program has two aliases, DFHWBTTB and DFHWBTTC.

DFHWBLT
The CICS Web bridge exit.

Unescaping function, DFHWBUN

DFHWBUN provides an unescaping function for data which has been transmitted to CICS in its escaped form, but which the application needs to manipulate in its unescaped form.

Exits

Two global user exit points are provided in CICS Web support for HTTP client requests:

XWBOPEN, HTTP client open exit
XWBOPEN is called during processing of an EXEC CICS WEB OPEN command, which is used by an application program to open a connection with a server. It is designed for use to specify proxy servers that should be used for HTTP requests by CICS as an HTTP client, and to apply a security policy to the host name specified for those requests.

XWBSNDO, HTTP client send exit
XWBSNDO is called during processing of an EXEC CICS WEB SEND or EXEC CICS WEB CONVERSE command. It is designed for use to specify a security policy for HTTP requests, in particular for the path component of the request.

The exits are described in the *CICS Internet Guide*.

Trace

The trace point IDs for this function are of the form WB xxxx. The trace levels are WB 1, WB 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Part 3. CICS domains

This part describes the domains into which CICS is organized, and the functions within these domains.

Apart from the application domain and the two catalog domains, each domain has one section describing it. The application domain consists of so many components that each component is described in a separate section, except for the two catalog domains that are described in the same section.

Sections are ordered alphabetically by domain ID for quick reference.

Chapter 70. Application domain (AP)

The principal components of the application domain are described under Chapter 2, “Application domain,” on page 7.

Application domain’s specific gates

Table 30 summarizes the application domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 30. Application domain’s specific gates

Gate	Trace	Function	XPI
ABAB	AP 0741 AP 0742	CREATE_ABEND_RECORD UPDATE_ABEND_RECORD START_ABEND INQUIRE_ABEND_RECORD TAKE_TRANSACTION_DUMP	NO NO NO NO
APAP	AP 0910 AP 0911	TRANSFER_SIT	NO
APCR	AP 4E00 AP 4E01	ESTIMATE_ALL ESTIMATE_CHANGED EXPORT_ALL EXPORT_CHANGED IMPORT_ALL IMPORT_CHANGED,	NO NO NO NO NO NO
APEX	AP 0510 AP 0515	INVOKE_USER_EXIT	NO
APID	AP 092A AP 092B	PROFILE QUERY_NETNAME	NO
APIQ	AP 0920 AP 0921	INQ_APPLICATION_DATA	YES
APJC	AP F900 AP F901	WRITE_JOURNAL_DATA	YES
APLH	AP 19A0 AP 19A1	ESTABLISH_LANGUAGE NOTIFY_REFRESH START_PROGRAM	NO NO NO
APLI	AP 1940 AP 1941	ESTABLISH_LANGUAGE START_PROGRAM	NO NO
APLJ	AP 1960 AP 1961	ESTABLISH_LANGUAGE START_PROGRAM PIPI_INIT_SUB_DP PIPI_CALL_SUB PIPI_TERM	NO NO NO NO NO
APRT	AP 1900 AP 1901	ROUTE_TRANSACTION	NO
APTC	AP 4900 AP 4901	CANCEL CLOSE EXTRACT_PROCESS ISSUE_NOTIFY LISTEN OPEN RECEIVE SEND SET_SESSION	NO NO NO NO NO NO NO NO NO
APTD	AP F600 AP F601	WRITE_TRANSIENT_DATA READ_TRANSIENT_DATA DELETE_TRANSIENT_DATA INITIALIZE_TRANSIENT_DATA RESET_TRIGGER_LEVEL	NO
APXM	AP 0590 AP 0591	INIT_XM_CLIENT BIND_XM_CLIENT RELEASE_XM_CLIENT	NO NO NO

Table 30. Application domain's specific gates (continued)

Gate	Trace	Function	XPI
BRAI	AP 4AE0 AP 4AE1	INQUIRE_AUTOINSTALL SET_AUTOINSTALL INSTALL_BRIDGE_FACILITY DELETE_BRIDGE_FACILITY	NO NO NO NO
BRAT	AP 2800 AP 2801	ATTACH	NO
BRFM	AP 2140 AP 2141	Subroutine for bridge facility allocation/deletion.	NO
BRFR	AP 4A00 AP 4A01	ALLOCATE_BRIDGE_FACILITY REALLOCATE_BRIDGE_FACILITY DETACH_BRIDGE_FACILITY SET_BRIDGE_FACILITY INQUIRE_BRIDGE_FACILITY STARTBR_BRIDGE_FACILITY GET_NEXT_BRIDGE_FACILITY ENDBR_BRIDGE_FACILITY GARBAGE_COLLECT	NO NO NO NO NO NO NO NO NO
BRIC	AP 2166 AP 2167	Subroutine interfacing interval control EXEC commands and the bridge exit.	NO
BRIQ	AP 2820	INQUIRE_CONTEXT	YES
BRLK	AP 4A20 AP 4A21	START_BRIDGE CONTINUE_BRIDGE ABEND_BRIDGE	NO NO NO
BRME	AP 4C40 AP 4C41	API_EMULATOR	NO
BRMF	AP 4C20 AP 4C21	FORMATTER	NO
BRMG	AP 4A40 AP 4A41	ALLOCATE_MESSAGE REALLOCATE_MESSAGE OUTPUT_MESSAGE RESEND_MESSAGE DELETE_MESSAGE READ_VECTOR ALLOCATE_VECTOR ERASE_OUTPUT_VECTORS	NO NO NO NO NO NO NO NO
BRMS	AP 2160 AP 2161	Subroutine interfacing BMS EXEC commands and the bridge exit.	NO
BRNS	AP 4A60 AP 4A61	INITIALISE_NUMBER CONNECT_NUMBER DISCONNECT_NUMBER ALLOCATE_NUMBER RELEASE_NUMBER	NO NO NO NO NO
BRRM	AP 2840 AP 2841	RMRO callback for PREPARE and COMMIT	NO
BRSP	AP 216C AP 216D	Subroutine interfacing syncpoint requests and the bridge exit.	NO
BRTC	AP 2163 AP 2164	Subroutine interfacing terminal control EXEC commands and the bridge exit.	NO
BRXM	AP 2860 AP 2861	XMAC callback for INIT_XM_CLIENT and BIND_XM_CLIENT	NO
ICXM	AP 05C0 AP 05C1	INQUIRE_FACILITY	NO
LILI	AP 1940 AP 1941	INITIALIZE_LANGUAGES TERMINATE_LANGUAGES FIND_PROGRAM_ATTRIBUTES GO_TO	NO NO NO NO
RTSU	AP 1910 AP 1911	COMMIT_SURROGATE FREE_SURROGATE GET_RECOVERY_STATUS PREPARE_SURROGATE RESET_SURROGATE	NO NO NO NO NO
TDOC	AP F640 AP F641	OPEN_TRANSIENT_DATA CLOSE_TRANSIENT_DATA CLOSE_ALL_EXTRA_TD_QUEUES	NO NO NO

Table 30. Application domain's specific gates (continued)

Gate	Trace	Function	XPI
TDTM	AP F680 AP F681	ADD_REPLACE_TDQDEF INQUIRE_TDQDEF START_BROWSE_TDQDEF GET_NEXT_TDQDEF END_BROWSE_TDQDEF SET_TDQDEF DISCARD_TDQDEF COMMIT_TDQDEFS	NO NO NO NO NO NO NO NO
TDXM	AP 05B0 AP 05B1	BIND_FACILITY BIND_SECONDARY_FACILITY RELEASE_FACILITY INQUIRE_FACILITY	NO NO NO NO
SAIQ	AP E120 AP E122	INQUIRE_SYSTEM SET_SYSTEM	YES YES
TFAL	AP D600 AP D601	ALLOCATE CANCEL_AID CHECK_TRANID_IN_USE CANCEL_AIDS_FOR_CONNECTION CANCEL_AIDS_FOR_TERMINAL DISCARD_AIDS FIND_TRANSACTION_OWNER GET_MESSAGE INITIALIZE_AID_POINTERS INQUIRE_ALLOCATE_AID LOCATE_AID LOCATE_REMDEL_AID LOCATE_SHIPPABLE_AID MATCH_TASK_TO_AID PURGE_ALLOCATE_AIDS RECOVER_START_DATA REMOTE_DELETE REMOVE_EXPIRED_AID REMOVE_EXPIRED_REMOTE_AID REMOVE_MESSAGE REMOVE_REMOTE_DELETES REROUTE_SHIPPABLE_AIDS RESCHEDULE_BMS RESET_AID_QUEUE RESTORE_FROM_KEYPOINT RETRIEVE_START_DATA SCHEDULE_BMS SCHEDULE_START SCHEDULE_TDP SLOWDOWN_PURGE TAKE_KEYPOINT TERM_AVAILABLE_FOR_QUEUE TERMINAL_NOW_UNAVAILABLE UNCHAIN_AID UPDATE_TRANNUM_FOR_RESTART	NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO
TFBF	AP 1730 AP 1731	BIND_FACILITY	NO
TFIQ	AP 1700 AP 1701	INQUIRE_TERMINAL_FACILITY INQUIRE_MONITOR_DATA SET_TERMINAL_FACILITY	NO NO NO
TFRF	AP 1710 AP 1711	RELEASE_FACILITY	NO
TFXM	AP 1790 AP 1791	INIT_XM_CLIENT BIND_XM_CLIENT	NO NO
MRXM	AP 17B0 AP 17B1	INIT_XM_CLIENT BIND_XM_CLIENT	NO NO
62XM	AP 17C0 AP 17C1	INIT_XM_CLIENT BIND_XM_CLIENT	NO NO

ABAB gate, CREATE_ABEND_RECORD function

The CREATE_ABEND_RECORD function of the ABAB gate is used to create an abend record (TACB).

Input parameters

[ABEND_CODE] is the four-character transaction abend code.

[FAILING_PROGRAM]
is the name of the program in which the abend occurred.

[REQUEST_ID] is the request ID from the TCTTE for a terminal-oriented task.

[FAILING_RESOURCE]
is the name of the system TCTTE (the connection) if the abend was raised by DFHZAND.

[REMOTE_SYSTEM]
is the name of the remote system if the abend was raised in the client transaction to reflect an abend occurring in the DPL server.

[SENSE_BYTES]
is the SNA sense bytes if the abend was raised by DFHZAND.

[ERROR_MESSAGE]
is the error message sent from the remote system if the abend was raised by DFHZAND.

[EXECUTION_KEY]
is a code indicating the execution key at the time the abend was issued, or at the time the operating system abend or program check occurred.

[STORAGE_TYPE]
is a code indicating the storage hit on an OC4.

[ERROR_OFFSET]
is the offset of a program check or operating system abend in the failing application program or CICS AP domain program.

[GENERAL_REGISTERS]
is the contents of the general purpose registers at the time of a program check or operating system abend.

[PSW] is the contents of the PSW at the time of a program check or operating system abend.

[INTERRUPT_DATA]
is the interrupt code and instruction length code etc, at the time of a program check or operating system abend.

[ALET] is the access list entry token (ALET) at the time of a program check or operating system abend.

[STOKEN] is the subspace token (STOKEN) at the time of a program check or operating system abend.

[SPACE] indicates whether the task was in SUBSPACE or BASESPACE mode at the time of a program check or operating system abend. It can have any of these values:

BASESPACE|SUBSPACE|NOSPACE

[GREG_ORDER] indicates the order of the registers passed in GENERAL_REGISTERS. DFHSRP saves the registers in the abend record in the order 0-15, and INQUIRE_ABEND_RECORD will always return them in this order. It can have either of these values:

R14TOR13|R0TOR15

[ACCESS_REGISTERS]
is the contents of the access registers at the time of a program check or operating system abend.

[FLOATING_POINT_REGISTERS]
is the contents of the floating point registers at the time of a program check or operating system abend.

[STATUS_FLAGS]
is the status flags at the time of the abend.

Output parameters

ABEND_TOKEN is the token allocated by ABAB for this abend. The token must be passed on subsequent UPDATE_ABEND_RECORD and START_ABEND requests to ABAB. The token is no longer valid after a START_ABEND request.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

ABAB gate, UPDATE_ABEND_RECORD function

The UPDATE_ABEND_RECORD function of the ABAB gate is used to update an abend record (TACB).

Input parameters

ABEND_TOKEN
is the token allocated by ABAB for this abend (on a preceding CREATE_ABEND_RECORD request). The token must be passed on subsequent UPDATE_ABEND_RECORD and START_ABEND requests to ABAB. The token is no longer valid after a START_ABEND request.

[ABEND_CODE]
is the four-character transaction abend code.

[FAILING_PROGRAM]
is the name of the program in which the abend occurred.

[REQUEST_ID]
is the request ID from the TCTTE for a terminal-oriented task.

[FAILING_RESOURCE]
is the name of the system TCTTE (the connection) if the abend was raised by DFHZAND.

[REMOTE_SYSTEM]
is the name of the remote system if the abend was raised in the client transaction to reflect an abend occurring in the DPL server.

[SENSE_BYTES]
is the SNA sense bytes if the abend was raised by DFHZAND.

[ERROR_MESSAGE]
is the error message sent from the remote system if the abend was raised by DFHZAND.

[EXECUTION_KEY]
is a code indicating the execution key at the time the abend was issued, or at the time the operating system abend or program check occurred.

[STORAGE_TYPE]
is a code indicating the storage hit on an OC4.

[ERROR_OFFSET]
is the offset of a program check or operating system abend in the failing application program or CICS AP domain program.

[GENERAL_REGISTERS]
is the contents of the general purpose registers at the time of a program check or operating system abend.

[PSW]
is the contents of the PSW at the time of a program check or operating system abend.

[INTERRUPT_DATA]
is the interrupt code and instruction length code etc, at the time of a program check or operating system abend.

[ALET]
is the access list entry token (ALET) at the time of a program check or operating system abend.

[STOKEN]
is the subspace token (STOKEN) at the time of a program check or operating system abend.

[SPACE]
indicates whether the task was in SUBSPACE or BASESPACE mode at the time of a program check or operating system abend. It can have any of these values:

`BASESPACE|SUBSPACE|NOSPACE`

[GREG_ORDER]
indicates the order of the registers passed in GENERAL_REGISTERS. DFHSRP saves the registers in the abend record in the order 0-15, and INQUIRE_ABEND_RECORD will always return them in this order. It can have either of these values:

`R14TOR13|R0TOR15`

[ACCESS_REGISTERS]
is the contents of the access registers at the time of a program check or operating system abend.

[FLOATING_POINT_REGISTERS]
is the contents of the floating point registers at the time of a program check or operating system abend.

[STATUS_FLAGS]
is the status flags at the time of the abend.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	INVALID_TOKEN

ABAB gate, START_ABEND function

The START_ABEND function of the ABAB gate is used to start transaction abend processing.

Input parameters

ABEND_TOKEN
is the token allocated by ABAB for this abend (on a preceding CREATE_ABEND_RECORD request).

[DUMP]
indicates whether a transaction dump should be produced for this abend. It can have either of these values:

YES|NO

[IGNORE_HANDLES]
indicates whether this abend should be passed to any EXEC CICS HANDLE routines that are active. IGNORE_HANDLES(YES) results in EXEC CICS HANDLE being ignored at all levels of the program stack. It can have either of these values:

YES|NO

Output parameters

RETRY_ADDRESS
if an XPCTA exit requests retry, control returns to the point of invocation of start_info, passing the retry address. This address includes the AMODE indicator in the first bit; it can be used as the target address in a DFHAM TYPE=BRANCH by the caller of START_ABEND GENERAL_REGISTERS is also set to point to the list of registers to be used for the retry, and SPACE to indicate the subspace.

[GENERAL_REGISTERS]
is the contents of the general purpose registers at the time of a program check or operating system abend.

[SPACE]
indicates whether the task was in SUBSPACE or BASESPACE mode at the time of a program check or operating system abend.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	INVALID_TOKEN

ABAB gate, INQUIRE_ABEND_RECORD function

The INQUIRE_ABEND_RECORD function of the ABAB gate is used to inquire about an abend record (TACB).

Input parameters

[ABEND_TYPE] indicates which abend record the information is to be extracted from. It can have any of these values:

LATEST|FIRST|LASTASRA

Output parameters

[ABEND_CODE] is the four-character transaction abend code.

[DUMP] indicates whether a dump was requested for this abend. It can have either of these values:

YES|NO

[REQUEST_ID] is the request ID from the TCTTE for a terminal-oriented task.

[FAILING_RESOURCE]
is the name of the system TCTTE (the connection) if the abend was raised by DFHZAND.

[FAILING_PROGRAM]
is the name of the program in which the abend occurred.

[REMOTE_SYSTEM]
is the name of the remote system if the abend was raised in the client transaction to reflect an abend occurring in the DPL server.

[SENSE_BYTES]
is the SNA sense bytes if the abend was raised by DFHZAND.

[ERROR_MESSAGE]
is the error message sent from the remote system if the abend was raised by DFHZAND.

[EXECUTION_KEY]
is a code indicating the execution key at the time the abend was issued, or at the time the operating system abend or program check occurred.

[STORAGE_TYPE]
is a code indicating the storage hit on an OC4.

[ERROR_OFFSET]
is the offset of a program check or operating system abend in the failing application program or CICS AP domain program.

[GENERAL_REGISTERS]
is the contents of the general purpose registers at the time of a program check or operating system abend.

[PSW] is the contents of the PSW at the time of a program check or operating system abend.

[INTERRUPT_DATA]
is the interrupt code and instruction length code etc, at the time of a program check or operating system abend.

[ALET] is the access list entry token (ALET) at the time of a program check or operating system abend.

[STOKEN] is the subspace token (STOKEN) at the time of a program check or operating system abend.

[SPACE] indicates whether the task was in SUBSPACE or BASESPACE mode at the time of a program check or operating system abend. It can have any of these values:

BASESPACE|SUBSPACE|NOSPACE

[ACCESS_REGISTERS]
is the contents of the access registers at the time of a program check or operating system abend.

[FLOATING_POINT_REGISTERS]
is the contents of the floating point registers at the time of a program check or operating system abend.

[STATUS_FLAGS]
is the status flags at the time of the abend.

[IGNORE_HANDLES]
indicates whether this abend should be passed to any EXEC CICS HANDLE routines that are active. IGNORE_HANDLES(YES) results in EXEC CICS HANDLE being ignored at all levels of the program stack. It can have either of these values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	NO_ABEND_RECORD

ABAB gate, TAKE_TRANSACTION_DUMP function

The TAKE_TRANSACTION_DUMP function of the ABAB gate is used to take a transaction dump.

Notes:

1. The TRANSACTION resource definition must specify dump and DUMP(YES) must be specified or defaulted on the associated START_ABEND call.
2. A transaction dump is not taken if any of the following is true:
   - The application is going to handle the abend; that is, there is an active handle at this level and IGNORE_HANDLES(NO) is specified or defaulted on the associated START_ABEND call.
   - The application is Language Environment/370 enabled, in which case the language interface deals with the abend.
   - A transaction dump is currently in progress.

Input parameters

None.

Output parameters

None.

APAP gate, TRANSFER_SIT function

The TRANSFER_SIT function of the APAP gate is used to transfer the address of DFHSIT to the AP domain after a GET_PARAMETERS call from this domain to the parameter manager domain.

Input parameters

SIT specifies the address and length of the system initialization table (DFHSIT).

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_ADDRESS INCONSISTENT_RELEASE
INVALID	INVALID_SIT_LENGTH INVALID_ADDRESS INVALID_FUNCTION

APCR gate, ESTIMATE_ALL function

The ESTIMATE_ALL function of the APCR gate is used to estimate the size of terminal input/output area (TIOA) needed to ship a channel.

Input parameters

COMMAND
is the type of API command that caused the channel to be shipped. COMMAND can have any of these values:

START_MRO|START_ISC|LINK|RETURN

[CHANNEL_NAME]
is the name of the channel.

[CHANNEL_TOKEN]
is a token referencing the channel.

Output parameters

BYTES_NEEDED
is the total size, in bytes, of the exported channel, including channel and container headers and the overall length of the data in the containers. This total includes all bytes for all containers.

[CHANNEL_TOKEN_OUT]
contains, if CHANNEL_NAME was specified on input, a token referencing the channel.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_ERROR

APCR gate, ESTIMATE_CHANGED function

The ESTIMATE_CHANGED function of the APCR gate is used to obtain the size of the channel data structure that will be used to ship the containers that have been modified since the IMPORT_ALL call. Only new, modified, or deleted containers are shipped, with deleted containers being shipped as container headers only.

This call must be supplied with a CONTAINER_LIST from an earlier IMPORT_ALL call. This list specifies all the containers that were created by the IMPORT_ALL call.

The output includes a NEW_CONTAINER_LIST that can be passed to a subsequent EXPORT_CHANGED call. This list specifies all the containers that have been modified or deleted since the IMPORT_ALL call, and that must therefore be exported by EXPORT_CHANGED.

Input parameters

CHANNEL_TOKEN
is a token referencing the channel.

COMMAND
is the type of API command that caused the channel to be shipped. COMMAND can have any of these values:

START_MRO|START_ISC|LINK|RETURN

CONTAINER_LIST
is a list of all the containers in the channel, obtained from an earlier IMPORT_ALL call.

Output parameters

BYTES_NEEDED
is the size, in bytes, of the channel data structure needed to transmit the containers that have been modified since the IMPORT_ALL call. This figure includes:
- The total size of any new containers in the channel, including the container headers and the data in the containers
- The total size of any modified containers in the channel, including the container headers and the data in the containers
- The total size of the container headers for any deleted containers in the channel

NEW_CONTAINER_LIST
is a list of all the containers in the channel that have been created, modified, or deleted since the last IMPORT_ALL call. This list must be passed to a subsequent EXPORT_CHANGED call.

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_ERROR

APCR gate, EXPORT_ALL function

The EXPORT_ALL function of the APCR gate is used to export the complete contents of a channel.

If a TERMINAL_TOKEN is supplied, terminal control is used to export the channel.

If an RS_TOKEN is supplied, the channel is exported from a listener region by request streams.

If a CORRELATION_ID is supplied, the channel is exported from an AOR by request streams.

The COMMAND keyword is used both to document the origin of the request in a trace, and to control whether an ID is used before or after the overall channel length.

Input parameters

CHANNEL_TOKEN
is a token referencing the channel to be exported.

COMMAND
is the type of API command that caused the channel to be exported. COMMAND can have any of these values:

`START_MRO|START_ISC|LINK|RETURN|SIBUS`

[CORRELATION_ID]
If CORRELATION_ID is specified, the channel is exported from an AOR by request streams. (RZTA SEND_REPLY is used.)

[RS_TOKEN]
is a token referencing the request stream with which the channel is associated. If RS_TOKEN is specified, the channel is exported from a listener region by request streams. (RZSO SEND_REQUEST is used).

[TERMINAL_TOKEN]
is a token referencing the terminal with which the channel is associated. If TERMINAL_TOKEN is specified, CICS terminal control is used to export the channel.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_ERROR TERMINAL_ERROR

CHANNEL_ERROR
indicates a failure in extracting the channel and container details.

TERMINAL_ERROR
indicates that a transmission error was detected either by terminal control or by request streams. If the error is detected by terminal control, TC_ABEND, TC_RESPONSE, and TC_SENSE provide additional terminal control diagnostic information. These keywords can be omitted for requests to export a channel using a request stream.

[TC_ABEND]
is the terminal control abend code.

[TC_RESPONSE]
is the terminal control response code.

[TC_SENSE]
is the terminal sense code.

APCR gate, EXPORT_CHANGED function

The EXPORT_CHANGED function of the APCR gate is used to return only those parts of a channel that have changed since IMPORT_ALL was issued.

Currently, only terminal control is supported on this call. Request streams are not supported.

EXPORT_CHANGED must be supplied with a CONTAINER_LIST built by an ESTIMATE_CHANGED call.

Currently, EXPORT_CHANGED always builds output in the form used by the LINK commands (length followed by ID).

Input parameters

CHANNEL_TOKEN
is a token referencing the channel to be exported.

CONTAINER_LIST
is a list of the containers in this channel that have been created, modified, or deleted since the last IMPORT_ALL call. This list should be obtained from the NEW_CONTAINER_LIST field of an ESTIMATE_CHANGED call.

COMMAND
is the type of API command that caused the channel to be exported. Currently, COMMAND can have only the following value:

`LINK`

TERMINAL_TOKEN
is a token referencing the terminal with which the channel is associated. CICS terminal control is used to export the channel.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_ERROR DATA_ERROR TERMINAL_ERROR

CHANNEL_ERROR
indicates a failure in extracting the channel and container details.

DATA_ERROR
indicates a failure in extracting the data from the channel's containers.

TERMINAL_ERROR
indicates that a transmission error was detected by CICS terminal control. If an error is detected by terminal control, TC_ABEND, TC_RESPONSE, and TC_SENSE provide additional terminal control diagnostic information.

[TC_ABEND]
is the terminal control abend code.

[TC_RESPONSE]
is the terminal control response code.

[TC_SENSE]
is the terminal sense code.

APCR gate, IMPORT_ALL function

The IMPORT_ALL function of the APCR gate is used to import the complete contents of a channel.

Typically, IMPORT_ALL creates the channel (and all its containers) into which the channel data is imported. However, if the CHANNEL_TOKEN_IN keyword is specified, IMPORT_ALL can import into an existing channel. In this case, the existing channel is typically empty, though this is not enforced.

If a TERMINAL_TOKEN is supplied, terminal control is used to import the channel.

If an RS_TOKEN is supplied, the channel is imported into an AOR by request streams.

If neither a TERMINAL_TOKEN nor an RS_TOKEN is supplied, the channel is imported into a listener region by request streams.

The COMMAND keyword is used both to document the origin of the request in a trace, and to control whether an ID is used before or after the overall channel length.

Input parameters

CHANNEL_TOKEN_IN
is a token referencing an existing channel into which the channel data is to be imported.

COMMAND
is the type of API command that caused the channel to be imported. COMMAND can have any of these values:

`START_MRO|START_ISC|LINK|RETURN|SIBUS`

[DATA_START]
is the position of the beginning of the channel data in the inbound TIOA.

[RS_TOKEN]
is a token referencing the request stream with which the channel to be imported is associated. If RS_TOKEN is specified, the channel is imported into an AOR by request streams. (RZTA RECEIVE_REQUEST is used.)

If neither TERMINAL_TOKEN nor RS_TOKEN is specifed, the channel is imported into a listener region by request streams. (RZSO RECEIVE_REPLY is used.) In this case, a CORRELATION_ID is returned for use with a subsequent EXPORT_ALL request.

[TERMINAL_TOKEN]
is a token referencing the terminal with which the channel to be imported is associated. CICS terminal control is used to import the channel.

If TERMINAL_TOKEN is specified, DATA_START must also be specified, to identify the position of the beginning of the channel data in the inbound TIOA.

If neither TERMINAL_TOKEN nor RS_TOKEN is specifed, the channel is imported into a listener region by request streams. (RZSO RECEIVE_REPLY is used.) In this case, a CORRELATION_ID is returned for use with a subsequent EXPORT_ALL request.

Output parameters

[CHANNEL_NAME]
is the name of the channel that has been created.

[CHANNEL_TOKEN]
is a token referencing the channel that has been created.

[CONTAINER_LIST]
is the address of a control block that identifies the initial state of the channel. It can be passed to a subsequent EXPORT_CHANGED call, when it is used to identify what changes have been made by comparing the initial state of the channel to the current state. This allows CICS to re-export only the changed containers.

[CORRELATION_ID]

[DATA_END]

[SIZE]

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DATA_ERROR TERMINAL_ERROR

DATA_ERROR
indicates that an error was detected in the inbound channel data. For example, an invalid channel or container header or a premature end to data all result in DATA_ERROR.

TERMINAL_ERROR
indicates that a transmission error was detected either by CICS terminal control or by request streams.

If an error is detected by terminal control, TC_ABEND, TC_RESPONSE, and TC_SENSE provide additional terminal control diagnostic information. These keywords can be omitted for requests to import a channel using a request stream.

[TC_ABEND]
is the terminal control abend code.

[TC_RESPONSE]
is the terminal control response code.

[TC_SENSE]
is the terminal sense code.

APCR gate, IMPORT_CHANGED function

The IMPORT_CHANGED function of the APCR gate is used to import those parts of a channel that have been modified since an EXPORT_ALL call. Any modified containers are either replaced or deleted. New containers are added. Unchanged containers are not received on the connection.

Currently, only terminal control intersystem links are supported. Request streams are not supported.

Currently, IMPORT_CHANGED always expects data in the format used for function-shipped LINK commands (length followed by ID).

DATA_START is used to locate the beginning of the channel data in the TIOA.

Input parameters

CHANNEL_TOKEN
is a token referencing the channel to be imported.

COMMAND
is the type of API command that caused the channel to be imported. COMMAND can have only the following value:

`LINK`

DATA_START
is the position of the beginning of the channel data in the inbound TIOA.

TERMINAL_TOKEN
is a token referencing the terminal with which the channel to be imported is associated. CICS terminal control is used to import the channel.

If TERMINAL_TOKEN is specified, DATA_START must also be specified, to identify the position of the beginning of the channel data in the inbound TIOA.

Output parameters

CHANNEL_ERROR
indicates a failure in extracting the channel and container details.

[DATA_END]
is the position in the final TIOA, immediately following the end of the channel.

[SIZE] The total number of bytes in all the containers in the channel.

RESPONSE
is the domain’s response to the call. It can have any of these values:

`OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_ERROR DATA_ERROR TERMINAL_ERROR

DATA_ERROR
indicates that an error was detected in the inbound channel data. For example, an invalid channel or container header or a premature end to data all result in DATA_ERROR.

TERMINAL_ERROR
indicates that a transmission error was detected either by CICS terminal control or by request streams.

If an error is detected by terminal control, TC_ABEND, TC_RESPONSE, and TC_SENSE provide additional terminal control diagnostic information. These keywords can be omitted for requests to import a channel using a request stream.

[TC_ABEND]
is the terminal control abend code.

[TC_RESPONSE]
is the terminal control response code.

[TC_SENSE]
is the terminal sense code.

APEX gate, INVOKE_USER_EXIT function

The INVOKE_USER_EXIT function of the APEX gate is used to invoke the user exit at a specified exit point.

Input parameters

EXIT_POINT
is the name of the exit.

TRACE
indicates whether or not user exits are to be traced. It can have either of these values:

YES|NO

[EXIT_PARAMETER_n]
is the parameter (number *n*) required by the exit. The nature of the parameter varies from one exit to another.

Output parameters

EXIT_RETURN_CODE
is the return code, if any, issued by the exit.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, DISASTER, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	EXIT_PROGRAM_FAILURE CHANGE_MODE_FAILURE
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION INVALID_EXIT_POINT

APID gate, PROFILE function

The PROFILE function of the APID gate extracts information from the AP domain profile for timeout.

Input parameters

NAME is the name of the profile

Output parameters

RTIMEOUT is the read timeout value.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	NOT_FOUND TM_LOCATE_FAILED

APID gate, QUERY_NETNAME function

The PROFILE function of the APID gate extracts information from the AP domain profile for timeout.

Input parameters

SYSID is the name of the sysid

Output parameters

NETNAME is the value of the netname for the given sysid.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	NOT_FOUND TM_LOCATE_FAILED

APIQ gate, INQ_APPLICATION_DATA function

The INQ_APPLICATION_DATA function of the APIQ gate is used to inquire about application data owned by the application domain.

Input parameters

None.

Output parameters

[EIB] is the address of the EXEC Interface Block.

[SYSEIB] is the address of the System EXEC Interface Block.

[TCTUA] is the address of the Task Control Table User Area.

[TCTUASIZE] is the length (in bytes) of the Task Control Table User Area.

[TWA] is the address of the Task Work Area.

[TWASIZE] is the length (in bytes) of the Task Work Area.

[RSA] is the address of the apllication's register save area.

[DSA] is the address of the head of the chain of dynamic storage for reentrant programs.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INQ_FAILED LOOP
EXCEPTION	DPL_PROGRAM NO_TRANSACTION_ENVIRONMENT TRANSACTION_DOMAIN_ERROR
INVALID	INVALID_FUNCTION

APJC gate, WRITE_JOURNAL_DATA function

The WRITE_JOURNAL_DATA function of the APJC gate is used to write a single record into a named journal.

Input parameters

JOURNALNAME is the journal identifier name.

JOURNAL_RECORD_ID
is the system type record identifier.

FROM is the address of the record.

[RECORD_PREFIX]
is the journal record user prefix.

WAIT specifies whether or not CICS is to wait until the record is written to auxiliary storage before returning control to the exit program. It can have either of these values:

YES|NO

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	JOURNAL_NOT_FOUND LENGTH_ERROR JOURNAL_NOT_OPEN STATUS_ERROR IO_ERROR
INVALID	INVALID_FORMAT INVALID_FUNCTION

APLH gate, ESTABLISH_LANGUAGE function

The ESTABLISH_LANGUAGE function of the APLH gate is used to establish the language of a compiled Java program with hot-pooling.

Input parameters

LOAD_POINT is the load point address of the program.

ENTRY_POINT is the entry point address of the program.

[PROGRAM_LENGTH]
is the length of the program.

[DEFINED_LANGUAGE]
is the language defined for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|LE370|PLI|NOT_DEFINED

EXECUTION_KEY
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

CICS|USER

DATA_LOCATION
defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). It can have either of these values:

ANY|BELOW

LANGUAGE_BLOCK
is a token identifying the current language block for the program.

PROGRAM
is the 8-character name of the program whose language is to be determined

REQUEST_TYPE
identifies the call of establish language. If the caller has a request type of link and establish language fails, then abend. Do not abend for a request type of load.

THREADSAFE
indicates whether whether the program is quasi-reentrant (and must execute on the QR TCB) or threadsafe (and can execute on the QR TCB or an OPEN TCB).

JVM_CLASS_PTR
is a token addressing the JVM class name length and value.

HOTPOOL
indicates whether the program is to be run in a hotpool under an H8 TCB, or under QR as usual.

JVM_PROG
indicates whether the request is for establish language for a JVM program.

Output parameters

[NEW_BLOCK]
is a new token identifying the new language block for the program.

[LANGUAGE_ESTABLISHED]
is the language established for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI|JVM|
ASSEMBLER_CICS|MVSLE370|
NOT_DEFINED|NOT_APPLIC

[CICSVAR_THREADSAFE]
is the threadsafe value established for the program. It can have any of these values:

YES|NO|NOT_DEFINED

[RUNTIME_ENVIRONMENT]
is the runtime environment established for the program. It can have any of these values:

JVM_RUNTIME|LE370_RUNTIME|NON_LE370_RUNTIME|
HOTPOOL_RUNTIME

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TRANSACTION_ABEND
INVALID	INVALID_FUNCTION

APLH gate, START_PROGRAM function

The START_PROGRAM function of the APLI gate is used to start a compiled Java program using hot-pooling.

Input parameters

PROGRAM
is the eight-character name of the program.

LINK_LEVEL
is the 16-bit value indicating the link-level of the program.

[CEDF_STATUS]
indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF). It can have any of these values:

CEDF|NOCEDF

[EXECUTION_SET]
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:

FULLAPI|DPLSUBSET|NOT_APPLIC

[PARMLIST_PTR]
is an optional token identifying the parameter list for the program.

COMMAREA
is an optional token identifying the communications area for the program.

[ENVIRONMENT_TYPE]
is the environment type of the program. It can have any of these values:

EXEC|GLUE|PLT|SYSTEM|TRUE|URM

[SYNCONRETURN]
defines whether or not a syncpoint is to be taken on return from the linked program. It can have either of these values:

YES|NO

LANGUAGE_BLOCK
LANGUAGE_BLOCK is a token giving the address of the current Program Language Block.

[SYSEIB_REQUEST]
indicates whether or not an EXEC CICS LINK or EXEC CICS XCTL had the SYSEIB translator option specified.

[DEFERRED_ABEND_FOR_XCTL]
indicates whether a Runaway type abend should be started on completion of the current START_PROGRAM.

Output parameters

ABEND_CODE
is the four-character abend code which is issued by CICS when an exception response is given and the cause of the error was a transaction abend.

IGNORE_PENDING_XCTL
indicates whether or not a pending XCTL should be ignored by program manager.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TRANSACTION_ABEND JVMPOOL_DISABLED JVM_PROFILE_NOT_FOUND JVM_PROFILE_NOT_VALID SYSTEM_PROPERTIES_NOT_FND USER_CLASS_NOT_FOUND
INVALID	INVALID_FUNCTION

APLH gate, NOTIFY_REFRESH function

The NOTIFY_REFRESH function is called to inform AP domain when a program is refeshed, so that it can quiesce all users of the program.

Input parameters

PROGRAM
is the eight-character name of the program.

Output parameters

[ABEND_CODE]
is the four-character abend code that is to be issued by CICS when an exception response is given and the cause of the error is a transaction abend.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TRANSACTION_ABEND
INVALID	INVALID_FUNCTION

APLI gate, ESTABLISH_LANGUAGE function

The ESTABLISH_LANGUAGE function of the APLI gate is used to establish the language of a conventional compiled program.

Input parameters

LOAD_POINT is the load point address of the program.

ENTRY_POINT is the entry point address of the program.

[PROGRAM_LENGTH]
is the length of the program.

[DEFINED_LANGUAGE]
is the language defined for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|LE370|PLI|NOT_DEFINED

EXECUTION_KEY
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

CICS|USER

DATA_LOCATION
defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). It can have either of these values:

ANY|BELOW

LANGUAGE_BLOCK
is a token identifying the current language block for the program.

PROGRAM is the 8-character name of the program whose language is to be determined

REQUEST_TYPE identifies the call of establish language. If the caller has a request type of link and establish language fails, then abend. Do not abend for a request type of load.

THREADSAFE indicates whether whether the program is quasi-reentrant (and must execute on the QR TCB) or threadsafe (and can execute on the QR TCB or an OPEN TCB).

JVM_CLASS_PTR
is a token addressing the JVM class name length and value.

HOTPOOL indicates whether the program is to be run in a hotpool under an H8 TCB, or under QR as usual.

JVM_PROG indicates whether the request is for establish language for a JVM program.

Output parameters

[NEW_BLOCK] is a new token identifying the new language block for the program.

[LANGUAGE_ESTABLISHED]
is the language established for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI|JVM|
ASSEMBLER_CICS|MVSLE370|
NOT_DEFINED|NOT_APPLIC

[CICSVAR_THREADSAFE]
is the threadsafe value established for the program. It can have any of these values:

YES|NO|NOT_DEFINED

[RUNTIME_ENVIRONMENT]
is the runtime environment established for the program. It can have any of these values:

JVM_RUNTIME|LE370_RUNTIME|NON_LE370_RUNTIME|
HOTPOOL_RUNTIME

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TRANSACTION_ABEND
INVALID	INVALID_FUNCTION

APLI gate, START_PROGRAM function

The START_PROGRAM function of the APLI gate is used to start a program.

Input parameters

PROGRAM
is the eight-character name of the program.

LINK_LEVEL
is the 16-bit value indicating the link-level of the program.

[CEDF_STATUS]
indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF). It can have any of these values:

CEDF|NOCEDF

[EXECUTION_SET]
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:

FULLAPI|DPLSUBSET|NOT_APPLIC

[PARMLIST_PTR]
is an optional token identifying the parameter list for the program.

COMMAREA
is an optional token identifying the communications area for the program.

[ENVIRONMENT_TYPE]
is the environment type of the program. It can have any of these values:

EXEC|GLUE|PLT|SYSTEM|TRUE|URM

[SYNCONRETURN]
defines whether or not a syncpoint is to be taken on return from the linked program. It can have either of these values:

YES|NO

LANGUAGE_BLOCK
LANGUAGE_BLOCK is a token giving the address of the current Program Language Block.

[SYSEIB_REQUEST]
indicates whether or not an EXEC CICS LINK or EXEC CICS XCTL had the SYSEIB translator option specified.

[DEFERRED_ABEND_FOR_XCTL]
indicates whether a Runaway type abend should be started on completion of the current START_PROGRAM.

Output parameters

ABEND_CODE
is the four-character abend code which is issued by CICS when an exception response is given and the cause of the error was a transaction abend.

IGNORE_PENDING_XCTL
indicates whether or not a pending XCTL should be ignored by program manager.

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TRANSACTION_ABEND JVMPOOL_DISABLED JVM_PROFILE_NOT_FOUND JVM_PROFILE_NOT_VALID SYSTEM_PROPERTIES_NOT_FND USER_CLASS_NOT_FOUND
INVALID	INVALID_FUNCTION

APLJ gate, ESTABLISH_LANGUAGE function

The ESTABLISH_LANGUAGE function of the APLI gate is used to establish the language parameters of a Java bytecode program.

Input parameters

LOAD_POINT
is the load point address of the program.

ENTRY_POINT
is the entry point address of the program.

[PROGRAM_LENGTH]
is the length of the program.

[DEFINED_LANGUAGE]
is the language defined for the program. It can have any of these values:

`ASSEMBLER|C370|COBOL|LE370|PLI|NOT_DEFINED`

EXECUTION_KEY
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

`CICS|USER`

DATA_LOCATION
defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). It can have either of these values:

`ANY|BELOW`

LANGUAGE_BLOCK
is a token identifying the current language block for the program.

PROGRAM
is the 8-character name of the program whose language is to be determined

REQUEST_TYPE
identifies the call of establish language. If the caller has a request type of link and establish language fails, then abend. Do not abend for a request type of load.

THREADSAFE
indicates whether whether the program is quasi-reentrant (and must execute on the QR TCB) or threadsafe (and can execute on the QR TCB or an OPEN TCB).

JVM_CLASS_PTR
is a token addressing the JVM class name length and value.

HOTPOOL
indicates whether the program is to be run in a hotpool under an H8 TCB, or under QR as usual.

JVM_PROG
indicates whether the request is for establish language for a JVM program.

Output parameters

[NEW_BLOCK] is a new token identifying the new language block for the program.

[LANGUAGE_ESTABLISHED]
is the language established for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI|JVM|
ASSEMBLER_CICS|MVSLE370|
NOT_DEFINED|NOT_APPLIC

[CICSVAR_THREADSAFE]
is the threadsafe value established for the program. It can have any of these values:

YES|NO|NOT_DEFINED

[RUNTIME_ENVIRONMENT]
is the runtime environment established for the program. It can have any of these values:

JVM_RUNTIME|LE370_RUNTIME|NON_LE370_RUNTIME|
HOTPOOL_RUNTIME

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TRANSACTION_ABEND
INVALID	INVALID_FUNCTION

APLJ gate, START_PROGRAM function

The START_PROGRAM function of the APLI gate is used to start a Java bytecode program.

Input parameters

PROGRAM is the eight-character name of the program.

LINK_LEVEL is the 16-bit value indicating the link-level of the program.

[CEDF_STATUS]
indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF). It can have any of these values:

CEDF|NOCEDF

[EXECUTION_SET]
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:

FULLAPI|DPLSUBSET|NOT_APPLIC

[PARMLIST_PTR]
is an optional token identifying the parameter list for the program.

COMMAREA is an optional token identifying the communications area for the program.

[ENVIRONMENT_TYPE]
is the environment type of the program. It can have any of these values:

EXEC|GLUE|PLT|SYSTEM|TRUE|URM

[SYNCONRETURN]
defines whether or not a syncpoint is to be taken on return from the linked program. It can have either of these values:

YES|NO

LANGUAGE_BLOCK
LANGUAGE_BLOCK is a token giving the address of the current Program Language Block.

[SYSEIB_REQUEST]
indicates whether or not an EXEC CICS LINK or EXEC CICS XCTL had the SYSEIB translator option specified.

[DEFERRED_ABEND_FOR_XCTL]
indicates whether a Runaway type abend should be started on completion of the current START_PROGRAM.

Output parameters

ABEND_CODE
is the four-character abend code which is issued by CICS when an exception response is given and the cause of the error was a transaction abend.

IGNORE_PENDING_XCTL
indicates whether or not a pending XCTL should be ignored by program manager.

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TRANSACTION_ABEND JVMPOOL_DISABLED JVM_PROFILE_NOT_FOUND JVM_PROFILE_NOT_VALID SYSTEM_PROPERTIES_NOT_FND USER_CLASS_NOT_FOUND
INVALID	INVALID_FUNCTION

APLJ gate, PIPI_INIT_SUB_DP function

The PIPI_INIT_SUB_DP function of the APLJ gate is used to invoke the Language Environment PIPI `init_sub_dp` function to initialize a subroutine environment.

Input parameters

PIPI_TABLE_ADDRESS
is the address of the PIPI table.

PIPI_SERVICE_RTNS
is the address of the PIPI service routine vector. For CICS, this addresses a vector providing entry points in DFHAPPIJ for GETSTORAGE and FREESTORAGE.

PIPI_RUNTIME_OPTIONS
is a character string containing the runtime options to be passed to Language Environment by PIPI.

EXECUTION_KEY
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

`CICS|USER`

Output parameters

PIPI_TOKEN
is a token identifying the PIPI subroutine environment for subsequent PIPI_CALL_SUB and PIPI_CALL_TERM calls.

[PIPI_RETURN_CODE]
is the return code set by PIPI.

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TRANSACTION_ABEND
INVALID	INVALID_FUNCTION

APLJ gate, PIPI_CALL_SUB function

The PIPI_CALL_SUB function of the APLJ gate is used to invoke the Language Environment PIPI `call_sub` function to call a subroutine in a previously initialized enclave.

Input parameters

PIPI_TOKEN
is the token identifying the PIPI subroutine environment, as returned on the PIPI_INIT_SUB_DP call.

PIPI_TABLE_INDEX
is the index into the PIPI table for the required function.

PIPI_CALL_PARAMETERS
is the address of the parameters to be passed on the PIPI subroutine call.

EXECUTION_KEY
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

CICS|USER

Output parameters

[PIPI_RETURN_CODE]
is the return code set by PIPI.

[PIPI_SUB_RETURN_CODE]
is the subroutine return code. If the enclave is terminated due to an unhandled condition, a STOP statement, or an EXIT statement (or an exit() function), this contains the return code for enclave termination.

[PIPI_SUB_REASON_CODE]
is the subroutine reason code. This is 0 for normal subroutine returns. If the enclave is terminated due to an unhandled condition, a STOP statement, or an EXIT statement (or an exit() function), this contains the reason code for enclave termination.

[PIPI_SUB_FEEDBACK]
is the feedback code for enclave termination. This is the CEE000 feedback code for normal subroutine returns. If the enclave is terminated due to an unhandled condition, a STOP statement, or an EXIT statement (or an exit() function), this contains the feedback code for enclave termination.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TRANSACTION_ABEND
INVALID	INVALID_FUNCTION

APLJ gate, PIPI_TERM function

The PIPI_TERM function of the APLJ gate is used to invoke the Language Environment PIPI `term` function to terminate a previously initialized enclave.

Input parameters

PIPI_TOKEN
is the token identifying the PIPI subroutine environment, as returned on the PIPI_INIT_SUB_DP call.

EXECUTION_KEY
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

CICS|USER

Output parameters

[PIPI_RETURN_CODE]
is the return code set by PIPI.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TRANSACTION_ABEND
INVALID	INVALID_FUNCTION

APRT gate, ROUTE_TRANSACTION function

The ROUTE_TRANSACTION function of the APRT gate is used to dynamically route transactions (which are defined to be dynamic and not automatically initiated) based on decisions made by the dynamic transaction routing program. For transactions which are automatically initiated or are defined to be remote and not dynamic, DFHAPRT will statically route such transactions.

Input parameters

DYNAMIC
indicates whether or not the transaction is defined as dynamic. It can have either of these values:

YES|NO

REMOTE
indicates whether or not the transaction is defined as remote. It can have either of these values:

YES|NO

REMOTE_NAME
is the four-character transaction identifier by which this transaction is to be known on the remote CICS region.

REMOTE_SYSTEM
is the eight-character name of the remote CICS region to which the transaction is to be routed.

DTRTRAN
indicates whether or not dynamic transaction routing is available. It can have either of these values:

YES|NO

Output parameters

RAN_LOCALLY
indicates whether or not the transaction ran on the local CICS region (that is, was not routed to a remote CICS region). It can have either of these values:

YES|NO

ABEND_CODE
is the four-character transaction abend code issued if the transaction terminates abnormally.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	PROGRAM_NOT_FOUND TRANSACTION_ABEND ISC_DISABLED REMOTE_CONN_OOS REMOTE_CONN_OOS_SYS_CHGD ALL_SESSIONS_BUSY ROUTE_FAILED DTRTRAN_REJECTED NOTAUTH

APTC gate, CANCEL function

The CANCEL function of the APTC gate invalidates the listening function.

Input parameters

TOKEN is the token for the session TCTTE

Output parameters

RESPONSE is the domain's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TOKEN_UNKNOWN TC_ERROR

APTC gate, CLOSE function

The CLOSE function of the APTC gate is used in cleanup.

Input parameters

TOKEN is the token for the session TCTTE

Output parameters

RESPONSE is the domain's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TOKEN_UNKNOWN TC_ERROR

APTC gate, EXTRACT_PROCESS function

The EXTRACT_PROCESS function of the APTC gate extracts information for the request.

Input parameters

NONE No input parameters

Output parameters

CONVID is the conversation id (which is the session tctte termid).

PROCESS_NAME is the name of the process to be invoked

SYNCLEVEL is the synclevel of the conversation

PIPDATA Applicable only for LU6.2 conversations

PIPDATA LENGTH
Applicable only for LU6.2 conversations.

RESPONSE is the domain's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	TOKEN_UNKNOWN TC_ERROR

APTC gate, ISSUE_NOTIFY function

The ISSUE_NOTIFY function of the APTC gate is used to inform the program DFHAPTC that a request has arrived.

Input parameters

TOKEN is the token for the session TCTTE

USER_TOKEN is the token supplied the the person who is to be notified.

ACTION specifies the action that should be taken.

Output parameters

RESPONSE is the domain's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TOKEN_UNKNOWN ISSUE_NOTIFY_ERROR ACTION_UNKNOWN

APTC gate, LISTEN function

The LISTEN function of the APTC gate is used to update the TCTTE with the user token.

Input parameters

TOKEN is the token for the session TCTTE

USER_TOKEN is a token supplied by requeststreams.

Output parameters

RESPONSE is the domain's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TOKEN_UNKNOWN TC_ERROR

APTC gate, OPEN function

The OPEN function of the APTC gate is used to allocate a session to the specified AOR.

Input parameters

SYSID specifies the name of the AOR

TRANID is the transaction name to be attached in the AOR.

NETNAME specifies the netname or applid of the AOR.

QUEUE is the queue option specified by the routing program.

Output parameters

ERROR CODE indicates the code passed back from the allocate procedure.

RESPONSE is the domain's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	OPEN_ERROR

APTC gate, RECEIVE function

The RECEIVE function of the APTC gate is used to receive data.

Input parameters

TOKEN is the token for the session TCTTE

RECEIVE_BUFFER
is the buffer into which the reply is to be placed.

Output parameters

LAST is an indicator to indicate if this is the last flow.

RESPONSE is the domain's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TOKEN_UNKNOWN RECEIVE_BUFFER_TOO_SMALL TC_ERROR NO_TCTTE

APTC gate, SEND function

The SEND function of the APTC gate is used to send the request to the AOR.

Input parameters

TOKEN is the token for the session TCTTE

SEND_BLOCK is the block data with the length and send data pointer.

PREFIX_AREA specifies the requeststreams information.

LAST is an indicator to indicate if this is the last flow.

Output parameters

RESPONSE is the domain's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TOKEN_UNKNOWN TC_ERRO NO_TCTTE

APTC gate, SET_SESSION function

The SET_SESSION function of the APTC gate is used to send the request to the AOR.

Input parameters

TOKEN is the token for the session TCTTE

RECOVERY_STATUS
indicates if recovery is necessary.

Output parameters

RESPONSE is the domain's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	TOKEN_UNKNOWN TC_ERROR

APTD gate, WRITE_TRANSIENT_DATA function

The WRITE_TRANSIENT_DATA function of the APTD gate is used to write a single record (or multiple records) to a named transient data queue.

Input parameters

QUEUE specifies the name of the queue to which the data is to be written

FROM_LIST is a list specifying the address and the length of each record that is to be written to the specified queue.

[RSL_CHECK] states whether resource-level checking is to be carried out. It can take the values:

YES|NO

Output parameters

[TD_RECORD] indicates the number of records that were successfully written to the transient data queue.

[TD_MIN_LENGTH]
indicates the minimum allowable length of a transient data record if a RESPONSE of EXCEPTION, and a REASON of LENGTH_ERROR is returned.

[TD_MAX_LENGTH]
indicates the maximum allowable length of a transient data record if a RESPONSE of EXCEPTION, and a REASON of LENGTH_ERROR is returned.

RESPONSE is Transient Data's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, DISASTER, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND CSM_ERROR DCT_ERROR DIRECTORY_MGR_ERROR LOGIC_ERROR
EXCEPTION	QUEUE_REMOTE QUEUE_NOT_FOUND QUEUE_NOT_AUTH QUEUE_DISABLED QUEUE_NOT_OPEN QUEUE_NOT_OUTPUT QUEUE_FULL NO_SPACE IO_ERROR LENGTH_ERROR LOCKED NO_RECOVERY_TABLE

RESPONSE	Possible REASON values
INVALID	INVALID_FROM_LIST_P INVALID_FROM_LIST_N INVALID_FROM_P INVALID_FROM_N INVALID_RSL_CHECK

APTD gate, READ_TRANSIENT_DATA function

The READ_TRANSIENT_DATA function of the APTD gate is used to read a single record from a named transient data queue.

Input parameters

QUEUE
specifies the name of the queue to which a record is to be read.

INTO
specifies a piece of storage into which the record is placed.

SUSPEND
specifies whether the caller wishes to wait if the record to be read has not been committed to the queue yet. It can take the values:

`YES|NO`

[RSL_CHECK]
states whether resource level checking is to be carried out. It can take the values:

`YES|NO`

[DATA_LOC]
if this is a READ TD SET rather than an INTO, DATA_LOC specifies whether Transient Data should obtain the required SET storage from above or below the 16MB line. It can take the values:

`ANY|BELOW`

[DATA_KEY]
if this is a READ TD SET rather than an INTO, DATA_KEY specifies whether Transient Data should obtain the required SET storage from CICS key or user key storage. It can take the values:

`CICS|USER`

Output parameters

RESPONSE
is Transient Data's response to the call. It can have any of the following values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND CSM_ERROR DCT_ERROR DIRECTORY_MGR_ERROR LOGIC_ERROR
EXCEPTION	QUEUE_REMOTE QUEUE_NOT_FOUND QUEUE_NOT_AUTH QUEUE_DISABLED QUEUE_NOT_OPEN QUEUE_NOT_INPUT QUEUE_BUSY IO_ERROR LENGTH_ERROR LOCKED

APTD gate, DELETE_TRANSIENT_DATA function

The DELETE_TRANSIENT_DATA function of the APTD gate is used to delete the specified transient data queue.

Input parameters

QUEUE
specifies the name of the queue to which the data is to be deleted.

[RSL_CHECK] states whether resource level checking is to be carried out. It can take the values:

YES|NO

[DISCARDING_DEFINITION]
states whether this DELETEQ request is part of an attempt by Transient Data to discard a transient data queue definition. It can take the values:

YES|NO

Output parameters

RESPONSE is Transient Data's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND CSM_ERROR DCT_ERROR DIRECTORY_MGR_ERROR LOGIC_ERROR
EXCEPTION	QUEUE_REMOTE QUEUE_NOT_FOUND QUEUE_NOT_AUTH QUEUE_DISABLED QUEUE_EXTRA IO_ERROR LOCKED NO_RECOVERY_TABLE

APTD gate, RESET_TRIGGER_LEVEL function

The RESET_TRIGGER_LEVEL function of the APTD gate is used to reset a transient data queue so that another trigger transaction can be attached. Sometimes it is necessary to include the RESET_TRIGGER_LEVEL function if a trigger transaction abends.

Input parameters

QUEUE specifies the name of the queue for which the trigger transaction is to be reset.

Output parameters

RESPONSE is Transient Data's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are: ABEND, DCT_ERROR, CSM_ERROR, DIRECTORY_MGR_ERROR, and LOGIC_ERROR.

APTD gate, INITIALISE_TRANSIENT_DATA function

The INITIALISE_TRANSIENT_DATA function of the APTD gate is invoked as part of the initialization process for the transient data facility.

Input parameters

None.

Output parameters

RESPONSE is Transient Data's response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND CSM_ERROR DCT_ERROR DIRECTORY_MGR_ERROR LOGIC_ERROR

APXM gate, TRANSACTION_INITIALIZATION function

The TRANSACTION_INITIALIZATION function of the APXM gate is called from the transaction manager domain to the AP Domain during transaction initialization. The AP domain allocates the AP domain transaction lifetime control blocks, and anchors them in the AP domains transaction token.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	GETMAIN_FAILURE

APXM gate, TRANSACTION_INITIALIZATION function

The TRANSACTION_INITIALIZATION function of the APXM gate is called from the transaction manager domain to the AP Domain during transaction initialization. The AP domain allocates the AP domain transaction lifetime control blocks, and anchors them in the AP domains transaction token.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	GETMAIN_FAILURE

APXM gate, RMI_START_OF_TASK function

The RMI_START_OF_TASK function of the APXM gate is called from transaction manager domain to the AP Domain during transaction initialization. The AP domain invokes any task-related user exits enabled for start of task.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

APXM gate, TRANSACTION_TERMINATION function

The TRANSACTION_TERMINATION function of the APXM gate is called from the transaction manager domain during transaction termination, and releases AP domain transaction lifetime resources.

Input parameters

TERMINATION_TYPE
is the type of transaction termination. It can have either of these values:

NORMAL|ABNORMAL

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	FREEMAIN_FAILURE

BRAI gate, INQUIRE_AUTOINSTALL function

Get the status of bridge autoinstall

Input parameters

None

Output parameters

AIBRIDGE
Values: **YES, AUTO**

Specifies if the URM is used (YES) or names are generated automatically (AUTO).

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON

BRAI gate, SET_AUTOINSTALL function

Set the status of bridge autoinstall

Input parameters

AIBRIDGE
Values: **YES, AUTO**

Specifies if the URM is used (YES) or names are generated automatically (AUTO).

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON

BRAI gate, INSTALL_BRIDGE_FACILITY function

Call the autoinstall URM to rename the bridge facility.

Input parameters

MECHANISM
Values: **LINK, START**

Specifies the type of mechanism used to invoke the bridge transaction. If invoked using the start bridge exit mechanism it is set to START and if invoked using the link3270 machanism it is set to LINK.

REQUESTED_TERMID
Termid requested by the client (if any).

REQUESTED_NETNAME
Netname requested by the client (if any).

Output parameters

TERMID Termid generated/accepted by the URM.

NETNAME Netname generated/accepted by the URM.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Values: **AUTOINSTALL_NOT_ACTIVE, LINK_URM_DISABLED, LINK_URM_FAILED, AUTOINSTALL_ABEND, INVALID_TERMID, INVALID_NETNAME, REQUEST_REJECTED**

BRAI gate, DELETE_BRIDGE_FACILITY function

Notify the URM that a bridge facility is being deleted.

Input parameters

MECHANISM Values: **LINK, START**

Specifies the type of mechanism used to invoke the bridge transaction. If invoked using the start bridge exit mechanism it is set to START and if invoked using the link3270 machanism it is set to LINK.

TERMID Termid generated/accepted by the URM.

NETNAME Netname generated/accepted by the URM.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Values: **AUTOINSTALL_NOT_ACTIVE, LINK_URM_DISABLED, LINK_URM_FAILED, AUTOINSTALL_ABEND**

BRAT gate, ATTACH function

The ATTACH function of the BRAT gate is called to attach a transaction with a bridge primary client.

Input parameters

TRANSACTION_ID
The 4 byte transaction id of the user transaction to be attached.

[BREXIT] An optional program name to be used as the bridge exit. If this is not specified, DFHBRAT will get the default value from transaction manager. If there is no default bridge exit, an error is returned.

[USERID] The USERID that should be signed-on to the terminal. This is only set when no facility token is passed.

[BRDATA] The address and length of a block of storage containing data to be passed to bridge exit. This is used as part of the primary client data.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_BREXIT NO_STORAGE USERID_NOT_AUTH_BREXIT NOT_FOUND DISABLED NO_XM_STORAGE NOT_ENABLED_FOR_SHUTDOWN STATE_SYSTEM_ATTACH
DISASTER	ABEND
INVALID	INVALID_FORMAT INVALID_FUNCTION

BRFR gate, ALLOCATE_BRIDGE_FACILITY function

Initial allocation of a session.

Input parameters

TRANSACTION Name of transaction on Router

USERID Userid of request

FACILITYKEEPTIME
Bridge facility expiry time

FACILITYLIKE Bridge facility model terminal

[REQUESTED_TERMID]
Client requested termid

[REQUESTED_NETNAME]
Client requested netname

Output parameters

FACILITYTOKEN
Bridge facilitytoken

TERMID Allocated Termid

NETNAME Allocated Netname

SEQNO Request sequence number

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Values: **NO_FREE_NAME, ALLOCATE_RANGE_FAILED, AUTOINSTALL_FAILED, AUTOINSTALL_BAD_TERMID, AUTOINSTALL_BAD_NETNAME, AUTOINSTALL_REJECTED**

BRFR gate, REALLOCATE_BRIDGE_FACILITY function

Reallocation of a session.

Input parameters

FACILITYTOKEN
Bridge facilitytoken

TRANSACTION Name of transaction on Router

USERID Userid of request

Output parameters

SYSID AOR owning bridge facility

FACILITYKEEPTIME
Bridge facility expiry time

FACILITYLIKE Bridge facility model terminal

[TERMID] Allocated Termid

[NETNAME] Allocated Netname

SEQNO Request sequence number

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
REASON Values: **INVALID_FACILITYTOKEN, FACILITYTOKEN_IN_USE, SECURITY_VIOLATION**

BRFR gate, DETACH_BRIDGE_FACILITY function

Detach a session. This will delete it if the keeptime is zero.

Input parameters

FACILITYTOKEN
Bridge facilitytoken
KEEP_INTERVAL
Bridge facility expiry time

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
REASON Values: **INVALID_FACILITYTOKEN, RELEASE_RANGE_FAILED**

BRFR gate, SET_BRIDGE_FACILITY function

Update the BFNB

Input parameters

FACILITYTOKEN
Bridge facilitytoken
[SYSID] AOR owning bridge facility
[REMOTE_TRANSACTION]
Name of transaction on AOR
[STATUS] Values: **RELEASED**

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
REASON
REASON Values: **INVALID_FACILITYTOKEN**

BRFR gate, INQUIRE_BRIDGE_FACILITY function

Get information in the BFNB.

Input parameters

FACILITYTOKEN
Bridge facilitytoken

Output parameters

[TERMID] Allocated Termid
[NETNAME] Allocated Netname
[TRANSACTION]
Name of transaction on Router
[TASKID] Taskid of request
[USERID] Userid of request
[SYSID] AOR owning bridge facility
[KEEP_INTERVAL]
Bridge facility expiry time
[STATUS] Values: **ACQUIRED, AVAILABLE, RELEASED**
State of bridge facility
RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Values: **INVALID_FACILITYTOKEN**

BRFR gate, STARTBR_BRIDGE_FACILITY function

Get a cursor for a BFNB browse.

Output parameters

BROWSE_TOKEN Browse cursor

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON

BRFR gate, GET_NEXT_BRIDGE_FACILITY function

Get information in a BFNB in a browse.

Input parameters

BROWSE_TOKEN Browse cursor

Output parameters

[FACILITYTOKEN]
Bridge facilitytoken

[TERMID] Allocated Termid

[NETNAME] Allocated Netname

[TRANSACTION]
Name of transaction on Router

[TASKID] Taskid of request

[USERID] Userid of request

[SYSID] AOR owning bridge facility

[KEEP_INTERVAL]
Bridge facility expiry time

[STATUS] Values: **ACQUIRED, AVAILABLE, RELEASED**

State of bridge facility

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Values: **BROWSE_END, INVALID_BROWSE_TOKEN**

BRFR gate, ENDBR_BRIDGE_FACILITY function

End a browse

Input parameters

BROWSE_TOKEN Browse cursor

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Values:

Invalid:

INVALID_BROWSE_TOKEN

BRFR gate, GARBAGE_COLLECT function

Get rid of any expired bridge facilities.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Values: **RELEASE_RANGE_FAILED**

BRIQ gate, INQUIRE_CONTEXT function

The INQUIRE_CONTEXT of the BRIQ gate is called to inquire on bridge state data.

Input parameters

[TRANSACTION_TOKEN]
The XM transaction token for the task to be inquired upon.

Output parameters

[CALL_EXIT_FOR_SYNC]
Can have either of these two values:

YES|NO

[BRIDGE_ENVIRONMENT]
Can have either of these two values:

YES|NO

[CONTEXT]
The transaction context. It can have either of these values:

NORMAL|BRIDGE|BREXIT

[START_CODE]
The emulated startcode of the user transaction

[BRIDGE_TRANSACTION_ID]
The transaction identifier of the bridge monitor (if CONTEXT is BRIDGE or BREXIT).

[BRIDGE_EXIT_PROGRAM]
The name of the bridge exit program (if CONTEXT is BRIDGE or BREXIT).

[BRIDGE_FACILITY_TOKEN]
A token identifying the bridge facility

[IDENTIFIER]
Data created by the bridge exit for problem determination purposes.

[BRDATA]
Data passed to the bridge exit during attach.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BAD_TOKEN NO_TRANSACTION_ENVIRONMENT
DISASTER	ABEND
INVALID	INVALID_FORMAT

BRLK gate, START_BRIDGE function

Start a transaction in a bridge environment.

Input parameters

TRANSACTION_ID
The id of the transaction to be started.

FACILITY_TOKEN
The bridge facility associated with this session.

[PRIORITY]
XM priority of the transaction

Output parameters

CONVERSATIONAL
Values: **YES, NO**

Is the bridge transaction in waiting.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Values: **TXN_ALREADY_RUNNING, NO_STORAGE, TRANSACTION_NOT_FOUND, DISABLED, NOT_ENABLED_FOR_SHUTDOWN, STATE_SYSTEM_ATTACH**

BRLK gate, CONTINUE_BRIDGE function

Connects to a bridge transaction to supply a reply to a conversional request.

Input parameters

FACILITY_TOKEN
The bridge facility associated with this session.

Output parameters

CONVERSATIONAL
Values: **YES, NO**

Is the bridge transaction in waiting.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Values: **TRANSACTION_NOT_RUNNING**

BRLK gate, ABEND_BRIDGE function

Connects to a bridge transaction waiting for a conversional request to tell it to abend.

Input parameters

FACILITY_TOKEN
The bridge facility associated with this session.

ABEND_CODE
The abend code with which to abend the bridge.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Values: **TRANSACTION_NOT_RUNNING**

BRME gate, API_EMULATOR function

Process a 3270 bridge exit request

Input parameters

BRXA
The bridge exit interface area

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Values: Internal error.

Exception:

BRMF gate, FORMATTER function

Process a 3270 bridge formatter request.

Input parameters

BRXA
The bridge exit interface area

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Values:

Exception

MESSAGE_ERROR

The Link3270 message contains a bad BRIV. Invalid

INVALID_FORMAT

Internal error.

INVALID_FUNCTION

Internal error.

INVALID_DATA

Internal error.

Exception:

BRMG gate, ALLOCATE_MESSAGE function

Allocate the message control blocks for a request.

Input parameters

FACILITYTOKEN
Bridge facilitytoken.

ADDRESS
Address of commarea.

INPUT_LENGTH
Length of input message.

OUTPUT_LENGTH
Maximum length of output message.

Output parameters

MESSAGE_TOKEN
Token representing the BMB instance.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Values: **INVALID_ADDRESS, INVALID_INPUT_LENGTH, INVALID_FACILITYTOKEN, INVALID_OUTPUT_LENGTH, STATE_ALREADY_ALLOC**

BRMG gate, REALLOCATE_MESSAGE function

Obtain access to message control blocks.

Input parameters

FACILITYTOKEN
Bridge facilitytoken.

ADDRESS
Address of commarea.

INPUT_LENGTH
Length of input message.

OUTPUT_LENGTH
Maximum length of output message.

UNRETURNED_VECTORS
Values: **CLEAR, KEEP**

Action to take on any unread BRIVs.

POSITION
Values: **IRRELEVANT, FIRST, SUBSEQUENT**

Should retrieve vectors be ignored?

Output parameters

MESSAGE_TOKEN
Token representing the BMB instance.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Values: **INVALID_ADDRESS, INVALID_FACILITYTOKEN, INVALID_INPUT_LENGTH, INVALID_OUTPUT_LENGTH, RETRIEVE_VECTOR_FOUND, BAD_VECTOR, STATE_NOT_ALLOC, STATE_ALREADY_ALLOC**

BRMG gate, OUTPUT_MESSAGE function

Get a copy of the outbound message. If insufficient room only copy complete vectors, and record how much is left.

Input parameters

MESSAGE_TOKEN
Token representing the BMB instance.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Values: **STATE_UNKNOWN_MSG, STATE_ALREADY_OUTPUT, INVALID_MESSAGE_TOKEN**

BRMG gate, RESEND_MESSAGE function

Get a copy of the previous outbound message.

Input parameters

FACILITYTOKEN
Bridge facilitytoken.

ADDRESS Address of commarea.

OUTPUT_LENGTH
Maximum length of output message.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Values: **INVALID_ADDRESS, INVALID_FACILITYTOKEN, INVALID_OUTPUT_LENGTH, STATE_NOT_OUTPUT, STATE_UNKNOWN_MSG, INVALID_MESSAGE_TOKEN**

BRMG gate, DELETE_MESSAGE function

Delete the message blocks.

Input parameters

MESSAGE_TOKEN
Token representing the BMB instance.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Values: **INVALID_FACILITYTOKEN, INVALID_MESSAGE_TOKEN**

BRMG gate, READ_VECTOR function

Read an inbound vector from the message. There are several vector types which each have have their own cursors. The vector address is valid until an OUTPUT_MESSAGE request.

Input parameters

MESSAGE_TOKEN
Token representing the BMB instance.

VECTOR_TYPE Values: **HEADER, RM, RE, CO, RT**

Output parameters

ADDRESS
Address of commarea.

VECTOR_LENGTH
Length of the vector allocated or read.

[ERROR_OFFSET]
For bad vectors, the offset of the field in error in the input message.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Values: **BAD_VECTOR, END_OF_MESSAGE, STATE_ALREADY_OUTPUT, STATE_UNKNOWN_MSG, INVALID_MESSAGE_TOKEN**

BRMG gate, ALLOCATE_VECTOR function

Allocate the storage for a new vector in the outbound message. The vector address is valid until an ALLOCATE_VECTOR or OUTPUT_MESSAGE request.

Input parameters

MESSAGE_TOKEN
Token representing the BMB instance.

VECTOR_TYPE
Values: **HEADER, BRIV**

[VECTOR_LENGTH]
Length of the vector allocated or read.

Output parameters

ADDRESS
Address of commarea.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Values: **STATE_ALREADY_OUTPUT, STATE_UNKNOWN_MSG, INVALID_MESSAGE_TOKEN**

BRMG gate, ERASE_OUTPUT_VECTORS function

Remove all BRIV's from the output message.

Input parameters

MESSAGE_TOKEN
Token representing the BMB instance.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Values: **STATE_ALREADY_OUTPUT, STATE_UNKNOWN_MSG, INVALID_MESSAGE_TOKEN**

BRNS gate, INITIALISE_NUMBER function

Initialise control blocks and the numberspace file.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
Values: **INVALID_FORMAT, INVALID_FUNCTION, ABEND, ADD_LOCK_FAILED, ADD_SUBPOOL_FAILED, ALREADY_INITIALISED, INQUIRE_KERNEL, UNLOCK_FAILED, UNKNOWN_KE_ERROR_CODE**

BRNS gate, CONNECT_NUMBER function

Connect to the numberspace, and build numberspace blocks

Input parameters

NUMBER_FILENAME
Numberspace file name.

NUMBER_SPACE Name of the numberspace.

OWNER

START_NUMBER The first number within the number range associated with the numberspace.

NUMBER_RANGE The range of numbers within the numberspace.

FREE_NUMBERS Values: **YES, NO**

Should numbers previously allocated to this region be freed during the first allocate number or release number request.

ENQUEUE Values: **YES, NO**

Are requests are enqueued or not. If YES and the allocated request has to wait an exception of enqueue_waiter is returned and a numbers is not allocated.

Output parameters

TOKEN

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] Values: **CONNECTED, NUMBER_FILE_ERROR, INVALID_FORMAT, INVALID_FUNCTION, ABEND, ADD_LOCK_FAILED, CONNECT_ERROR, GETMAIN_ERROR, INVALID_RANGE_NUMBER, INVALID_START_NUMBER, LOCK_FAILED, NOT_INITIALISED, UNKNOWN_KE_ERROR_CODE, UNLOCK_FAILED**

BRNS gate, DISCONNECT_NUMBER function

Disconnect from the numberspace. The numberspace blocks are freed and the local allocation record is deleted. If no numbers are allocated the C1 and C2 records are also deleted. The file is closed.

Input parameters

TOKEN

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] Values: **NUMBER_FILE_ERROR, INVALID_FORMAT, INVALID_FUNCTION, ABEND, ALLOC_RECORD_CORRUPT, C1_ALLOC_MISMATCH, C1_C2_MISMATCH, C1_RECORD_CORRUPT, C2_RECORD_CORRUPT, FREEMAIN_ERROR, INVALID_TOKEN, LOCK_FAILED, NOT_INITIALISED, NSBLK_C1_MISMATCH, NUMBERS_ALLOCATED, UNEXPECTED_NUMFILE_ERROR, UNKNOWN_KE_ERROR_CODE, UNKNOWN_RECORD_TYPE, UNLOCK_FAILED**

BRNS gate, ALLOCATE_NUMBER function

Allocate the next available number.

Input parameters

TOKEN

Output parameters

NUMBER Number allocated by the numberspace.

[CONNECTION_NUMBER]
The numberspace token. The connection number allocated to this regio

[NUMBER_RANGE]
The range of numbers within the numberspace.

[ALLOCATED_NUMBERS]
The allocated numbers within the range.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] Values: **NUMBER_FILE_ERROR, ALL_NUMBERS_ALLOCATED, ENQUEUE_WAITER, INVALID_FORMAT, INVALID_FUNCTION, ABEND, ALLOC_RECORD_CORRUPT, C1_ALLOC_MISMATCH, C1_C2_MISMATCH, C1_RECORD_CORRUPT, C2_RECORD_CORRUPT, INVALID_TOKEN, NOT_INITIALISED, NSBLK_C1_MISMATCH, UNEXPECTED_NUMFILE_ERROR, UNKNOWN_KE_ERROR_CODE, UNKNOWN_RECORD_TYPE, LOCK_FAILED, UNLOCK_FAILED**

BRNS gate, RELEASE_NUMBER function

Release an allocated number from the numberspace.

Input parameters

TOKEN

NUMBER Number allocated by the numberspace.

Output parameters

[CONNECTION_NUMBER]
The numberspace token. The connection number allocated to this region.

[NUMBER_RANGE]
The range of numbers within the numberspace.

[ALLOCATED_NUMBERS]
The allocated numbers within the range.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] Values: **NUMBER_FILE_ERROR, INVALID_FORMAT, INVALID_FUNCTION, ABEND, ALLOC_RECORD_CORRUPT, C1_ALLOC_MISMATCH, C1_C2_MISMATCH, C1_RECORD_CORRUPT, C2_RECORD_CORRUPT, INVALID_NUMBER, INVALID_TOKEN, NOT_INITIALISED, NSBLK_C1_MISMATCH, NUMBER_NOT_ALLOCATED, NUMBER_NOT_ALLOC_LOCALLY, UNEXPECTED_NUMFILE_ERROR, UNKNOWN_KE_ERROR_CODE, UNKNOWN_RECORD_TYPE, LOCK_FAILED, UNLOCK_FAILED**

ICXM gate, INQUIRE_FACILITY function

The INQUIRE_FACILITY function of the ICXM gate is used to inquire about the interval control facilities that support facility management calls from the transaction management domain.

Input parameters

[FACILITY_TOKEN]
is the token identifying the transaction that has been trigger-level attached.

Output parameters

FACILITY_NAME
is the four-character name of the transaction that has been trigger-level attached.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

LILI gate, INITIALIZE_LANGUAGES function

The INITIALIZE_LANGUAGES function of the LILI gate is called during CICS initialization to initialize Language Environment (Partition Initialization).

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION

LILI gate, TERMINATE_LANGUAGES function

The TERMINATE_LANGUAGES function of the LILI gate is called during CICS shutdown to terminate Language Environment (Partition Termination).

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION

LILI gate, FIND_PROGRAM_ATTRIBUTES function

The FIND_PROGRAM_ATTRIBUTES function of the LILI gate is called by EDF to find the working storage, static storage and entry point of a program. It issues a Determine_Working_Storage request to Language Environment.

Input parameters

USERS_RSA_POINTER
is the address of the user program's register save area.

Output parameters

WORKING_STORAGE
defines the area into which the address and length of Working Storage are to be put.

STATIC_STORAGE
defines the area into which the address and length of Static Storage are to be put.

PROGRAM_ENTRY_POINT
is a token giving the address of a location set by Language Environment to hold the true entry point of the program. This is the entry point of the program as seen by the user, and ignores any CICS and Language Environment stubs.

[ENTRY_POINT_NAME]
is a token giving the address of a location set by Language Environment to address a character string built by Language Environment that identifies the program at Program Entry Point. The area addressed is a halfword length value, followed by the character string, which has a maximum length of 64 characters.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION

LILI gate, GO_TO function

The GO_TO function of the LILI gate is used in the processing of the HANDLE command, when a condition or abend occurs, to produce information for the call to Language Environment's Perform_goto function.

Input parameters

LANGUAGE
is the language of the program that issued the HANDLE command, as taken from the handle table.

LABEL
For COBOL languages, this is the address of a special Register Save Area built by CICS, which contains the values of the registers at the time the original HANDLE ABEND command was issued. Language Environment is able to determine the GO_TO address from values set in this RSA. For other languages, this is the actual GO_TO address.

USERS_RSA
is the address of the Register Save Area at the time of the HANDLE command.

Output parameters

PARAMETER_LIST
is the address of a parameter list built by DFHLILI, conforming to the Extended Run Time Language Interface protocols, which is to be used when calling Language Environment to drive GO_TO.

REGISTER_SAVE_AREA
is the address of a save area, provided by DFHLILI, for use by Language Environment during the processing of the GO_TO call.

ENVIRONMENT_ENTRY_POINT
is the address of the entry point to the Language Environment interface module CEECCICS.

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION

TFXM gate, INIT_XM_CLIENT function

The INIT_XM_CLIENT function of the TFXM gate is the initialization phase of the transaction initialization that has been initiated from a terminal or an LU6.1 session.

Input parameters

[PRIMARY_CLIENT_BLOCK]
is the address of the TCTTE and its length.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

TFXM gate, BIND_XM_CLIENT function

The BIND_XM_CLIENT function of the TFXM gate is the bind phase of the transaction initialization that has been initiated from a terminal or an LU6.1 session.

Input parameters

[PRIMARY_CLIENT_BLOCK]
is the address of the TCTTE and its length.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

MRXM gate, INIT_XM_CLIENT function

The INIT_XM_CLIENT function of the MRXM gate is the initialization phase of the transaction initialization that has been initiated from a terminal or an MRO session.

Input parameters

[PRIMARY_CLIENT_BLOCK]
is the address of the TCTTE and its length.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

MRXM gate, BIND_XM_CLIENT function

The BIND_XM_CLIENT function of the MRXM gate is the bind phase of the transaction initialization that has been initiated from a terminal or an MRO session.

Input parameters

[PRIMARY_CLIENT_BLOCK]
is the address of the TCTTE and its length.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

62XM gate, INIT_XM_CLIENT function

The INIT_XM_CLIENT function of the 62XM gate is the initialization phase of the transaction initialization that has been initiated from a terminal or an LU6.2 or APPC session.

Input parameters

[PRIMARY_CLIENT_BLOCK]
is the address of the TCTTE and its length.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

62XM gate, BIND_XM_CLIENT function

The BIND_XM_CLIENT function of the 62XM gate is the bind phase of the transaction initialization that has been initiated from a terminal or an LU6.2 or APPC session.

Input parameters

[PRIMARY_CLIENT_BLOCK]
is the address of the TCTTE and its length.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

RTSU gate, COMMIT_SURROGATE function

The COMMIT_SURROGATE function of the RTSU gate is used to update the state of a surrogate TCTTE when a Unit of Work is committed or backed out.

Input parameters

SURROGATE
The address of the surrogate TCTTE

[UOW_STATUS]
Indicates if the Unit of Work is being committed or backed out. It can have either of these two values:
FORWARD|BACKWARD

Output parameters

FREE_REQUIRED
Indicates if the surrogate should now be freed (because, for instance, the relay link has been freed). It can have either of these two values:
YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION INVALID_SURROGATE INVALID_SAVED_STATE

RTSU gate, FREE_SURROGATE function

The FREE_SURROGATE function of the RTSU gate is used to free a surrogate TCTTE from the currently executing task.

Input parameters

SURROGATE The address of the surrogate TCTTE

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
INVALID	INVALID_FORMAT INVALID_FUNCTION INVALID_SURROGATE

RTSU gate, GET_RECOVERY_STATUS function

The GET_RECOVERY_STATUS function of the RTSU gate is used to determine what actions are required of the relay link at syncpoint.

Input parameters

SURROGATE The address of the surrogate TCTTE

Output parameters

RECOVERY_STATUS
Indicates the syncpoint protocols required on the relay link. It can have any of these values:

NECESSARY|UNNECESSARY|SYNC_LEVEL_1

ABORT_ALLOWED
Indicates whether, during the syncpoint protocols, an ABORT FMH7 should be sent on the relay link. It can have either of these values:

YES|NO

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
INVALID	INVALID_FORMAT INVALID_FUNCTION INVALID_SURROGATE

RTSU gate, PREPARE_SURROGATE function

The PREPARE_SURROGATE function of the RTSU gate is used to update the state of a surrogate TCTTE at the start of syncpoint.

Input parameters

SURROGATE The address of the surrogate TCTTE

INITIATOR Indicates if the associated relay link is the initiator of the syncpoint request. It can have either of these two values:

YES|NO

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_STATE
DISASTER	ABEND
INVALID	INVALID_FORMAT INVALID_FUNCTION INVALID_SURROGATE

RTSU gate, RESET_SURROGATE function

The RESET_SURROGATE function of the RTSU gate is used to restore the state of a surrogate TCTTE when ISSUE_ABEND or ISSUE_ERORR was received on the relay link in reply to an ISSUE PREPARE request.

Input parameters

SURROGATE The address of the surrogate TCTTE

REPLY_TO_PREPARE
Indicates which reply was received in response to ISSUE_PREPARE. It can have either of these two values:

ISSUE_ERROR¬ISSUE_ABEND

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
INVALID	INVALID_FORMAT INVALID_FUNCTION INVALID_SURROGATE

SAIQ gate, INQUIRE_SYSTEM function

The INQUIRE_SYSTEM function of the SAIQ gate is used to inquire about system data owned by the application domain.

Input parameters

[GMMTEXT] is an optional token identifying the text of the "good-morning" message.

Output parameters

[AKP] is a fullword binary field indicating the activity keypoint frequency, in the range 200 through 65 535, of the local CICS region.

[CICSREL] is a 4-character string indicating the level (version and release numbers) of CICS code present.

[CICSSTATUS] is the current status of the local CICS system. It can have any of these values:
ACTIVE|FIRSTQUIESCE|FINALQUIESCE|INITIALIZING

[CICSSYS] is the one-character identifier of the operating system for which the running CICS system has been built. A value of “X” represents MVS system with extended addressing.

[CWA] is the address of the CWA.

[CWALENGTH] is the length (in bytes) of the CWA.

[DATE] is a four-character packed-decimal value indicating the current date (00yydddc, where yy=years, ddd=days, c is the sign).

[DCE_SUFFIX] is the two-character suffix of the DCE initialization side file, as specified on the DCESUFFX system initialization parameter.

[DTRPRGRM] is the 8-character name of the program controlling the dynamic routing of transactions.

[GMMLENGTH] is a halfword binary field indicating the length of the “good-morning” message text.

[GMMTRANID] is the four-character identifier of the “good-morning” transaction.

[INITSTATUS] is the initialization status of the local CICS region. It can have any of these values:
FIRSTINIT|SECONDINIT|THIRDINIT|INITCOMPLETE

[JOBNAME] is the eight-character MVS job name for the local CICS region.

[OPREL] indicates the release number of the operating system currently running. The values is ten times the formal release number. For example, “21” represents Release 2.1.

[OPSYS] is a one-character identifier indicating the type of operating system currently running. A value of “X” represents MVS.

[PLTPI] is the two-character suffix of the program list table, which contains a list of programs to be run in the final stages of system initialization.

[SECURITYMGR]
indicates whether an external security manager (such as RACF®) is active in the CICS region, or whether no security is being used. It can have either of these values:
EXTSECURITY|NOSECURITY

[SHUTSTATUS] is the shutdown status of the local CICS region. It can have any of these values:
CONTROLSHUT|SHUTDOWN|CANCELLED|NOTSHUTDOWN

[STARTUP] is the type of startup used for the local CICS region. It can have any of these values:
COLDSTART|WARMSTART|EMERGENCY|LOGTERM|STANDBY|AUTOSTART

[STARTUPDATE]
is a four-character packed-decimal value indicating the date on which the local CICS region was started.

[TERMURM] is the eight-character name of the terminal autoinstall program.

[TIMEOFDAY] is a four-character packed-decimal value indicating the time at which the local CICS region was started (hhmmsstc, where hh=hours, mm=minutes, ss=seconds, c is the sign).

[XRFSTATUS] indicates whether the local CICS region is a PRIMARY (active) or TAKEOVER (alternate) XRF CICS region, or has no XRF support. It can have any of these values:
PRIMARY|TAKEOVER|NOXRF

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INQ_FAILED LOOP
EXCEPTION	LENGTH_ERROR UNKNOWN_DATA

SAIQ gate, SET_SYSTEM function

The SET_SYSTEM function of the SAIQ gate is used to set system data values owned by the application domain.

Input parameters

[AKP] is a fullword binary field indicating the activity keypoint frequency, in the range 200 through 65 535, of the local CICS region.

[DCE_SUFFIX] is the two-character suffix of the DCE initialization side file.

[DTRPRGRM] is the 8-character name of the program controlling the dynamic routing of transactions.

[GMMTEXT] is an optional token identifying the text of the “good-morning” message.

[GMMLENGTH] is a halfword binary field indicating the length of the “good-morning” message text.

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND SET_FAILED LOOP
EXCEPTION	AKP_SIZE_ERROR LENGTH_ERROR NO_KEYPOINTING

TDOC gate, OPEN_TRANSIENT_DATA function

The OPEN_TRANSIENT_DATA function of the TDOC gate is used to open an extrapartition transient data queue.

Input parameters

QUEUE specifies the name of the extrapartition transient data queue to be opened.

TD_QUEUE_TOKEN
can be specified instead of QUEUE. The token uniquely identifies the extrapartition queue to be opened.

Output parameters

RESPONSE is Transient Data’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	QUEUE_INTRA QUEUE_REMOTE QUEUE_OPEN QUEUE_NOT_FOUND
DISASTER	DCT_ERROR DIRECTORY_MGR_ERROR LOGIC_ERROR

TDOC gate, CLOSE_TRANSIENT_DATA function

The CLOSE_TRANSIENT_DATA function of the TDOC gate is used to close an extrapartition transient data queue.

Input parameters

QUEUE specifies the name of the extrapartition transient data queue to be closed.

TD_QUEUE_TOKEN
can be specified instead of QUEUE. The token uniquely identifies the extrapartition queue to be closed.

Output parameters

RESPONSE
is Transient Data's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND DCT_ERROR DIRECTORY_MGR_ERROR LOGIC_ERROR
EXCEPTION	QUEUE_INTRA QUEUE_REMOTE QUEUE_CLOSED QUEUE_NOT_FOUND QUEUE_NULL QUEUE_NOT_CLOSED

TDOC gate, CLOSE_ALL_EXTRA_TD_QUEUES function

The CLOSE_ALL_EXTRA_TD_QUEUES function of the TDOC gate closes all extrapartition transient data queues which are currently open in the system. The CLOSE_ALL_EXTRA_TD_QUEUES function is usually invoked as part of a warm shutdown.

Input parameters

None.

Output parameters

RESPONSE
is Transient Data's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are: ABEND, DCT_ERROR, DIRECTORY_MGR_ERROR, and LOGIC_ERROR.

TDTM gate, ADD_REPLACE_TDQUEUE function

The ADD_REPLACE_TDQUEUE function of the TDTM gate is used to install a transient data queue definition.

Input parameters

QUEUE_NAME
specifies the name of the queue to be installed.

TD_QUEUE_TOKEN
can be specified instead of QUEUE. The token uniquely identifies a DCT entry that has already been built, but needs to be installed.

TD_TYPE
specifies the queue type. Possible values are:

EXTRA|INTRA|INDIRECT|REMOTE

BLOCK_LENGTH
specifies the block length of an extrapartition queue.

BUFFER_NUMBER
specifies the number of buffers to be associated with an extrapartition queue.

DDNAME
specifies the DDNAME to be associated with an extrapartition queue.

DISPOSITION
specifies the disposition of the data set to be associated with an extrapartition queue. Possible values are:

SHR|OLD|MOD

DSNAME
specifies the DSNAME of the data set to be associated with an extrapartition queue.

ERROR_OPTION
specifies the action to be taken in the event of an I/O error. This input parameter applies to extrapartition queues only. Possible values are:

IGNORE|SKIP

FACILITY
specifies the facility associated with this intrapartition queue when a trigger transaction is attached. Possible values are:

TERMINAL|FILE|SYSTEM

FACILITY_ID specified together with the FACILITY option, FACILITY_ID identifies the facility that the trigger transaction should be associated with.

INDIRECT_DEST
specifies the destination queue if this queue is an indirect queue.

WAIT_ACTION specifies the action to be taken if this logically recoverable intrapartition queue suffers an indoubt failure. Possible values are:

QUEUE|REJECT

WAIT specifies whether this logically recoverable intrapartition queue can wait for the resolution of an indoubt failure. Possible values are:

YES|NO

OPEN_TIME specifies whether this extrapartition queue should be opened as part of installation processing. Possible values are:

INITIAL|DEFERRED

RECORD_LENGTH
specifies the record length of an extrapartition queue in bytes.

RECORD_FORMAT
specifies the format of records held in an extrapartition queue. Possible values are:

FIXUNB|FIXUNBA|FIXUNBM|FIXBLK|FIXBLKA|FIXBLKM|
VARBLK|VARBLKA|VARBLKM|VARUNB|VARUNBA|
VARUNBM|UNDEFINED

RECOVERY specifies the recovery type of an intrapartition queue. Possible values are:

NO|PH|LG

REMOTE_NAME specifies the remote name of the queue if this is a remote queue definition.

REMOTE_SYSTEM
specifies the remote system identifier (SYSID) if this is a remote queue definition.

REWIND specifies where the tape is positioned in relation to the end of the data set. This input parameter applies to extrapartition queues only. Possible values are:

REREAD|LEAVE

TRANSACTION_ID
specifies the ATI transaction to be invoked when the trigger level is reached.

TRIGGER_LEVEL
specifies the trigger level of the intrapartition queue.

TYPE_FILE indicates whether this queue is:
- An input queue
- An output queue
- Whether the queue is to be read backwards.

Possible values are:

INPUT|OUTPUT|RDBACK

USERID specifies the userid to be associated with a trigger-level attached transaction.

SYSOUTCLASS specifies the SYSOUT class to be used for the associated output extrapartition queue.

Output parameters

RESPONSE is Transient Data's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND CATALOG_WRITE_FAILED DCT_ERROR DIRECTORY_MGR_ERROR LOGIC_ERROR

RESPONSE	Possible REASON values
EXCEPTION	COLD_START_IN_PROGRESS DDNAME_NOT_FOUND DFHINTRA_NOT_OPENED DISABLE_PENDING DUPLICATE INSUFFICIENT_STORAGE NOT_CLOSED NOT_DISABLED NOT_EMPTY NOT_SAME_TYPE QUEUE_NOT_OPENED SECURITY_FAILURE USERID_NOTAUTHED

TDTM gate, INQUIRE_TDQUEUE function

The INQUIRE_TDQUEUE function of the TDTM gate is used to inquire on a specified queue.

Input parameters

QUEUE_NAME
specifies the name of the queue to be inquired upon.

Output parameters

[ATI_FACILITY]
specifies the facility associated with this intrapartition queue when a trigger transaction is attached. Possible values are:

TERMINAL|FILE|SYSTEM

[ATI_TERMID]
specified together with the FACILITY option, FACILITY_ID identifies the facility that the trigger transaction should be associated with.

[ATI_TRANID]
specifies the ATI transaction to be invoked when the trigger level is reached.

[BUFFER_NUMBER]
specifies the number of buffers to be associated with an extrapartition queue.

[DDNAME]
specifies the DDNAME to be associated with an extrapartition queue.

[DISPOSITION]
specifies the disposition of the data set to be associated with an extrapartition queue. Possible values are:

SHR|OLD|MOD

[DSNAME]
specifies the DSNAME of the data set to be associated with the extrapartition queue.

[EMPTY_STATUS]
indicates whether the queue contains any records, and whether the queue is full. This option applies to extrapartition queues only. Possible values are:

FULL|EMPTY|NOTEMPTY

[ENABLE_STATUS]
indicates the status of the queue. Possible values are:

ENABLED|DISABLING|DISABLED

[ERROR_OPTION]
specifies what action is to be taken in the event of an I/O error. This option applies to extrapartition queues only. Possible values are:

IGNORE|SKIP

[INDIRECT_DEST]
specifies the destination queue if this queue is an indirect queue.

[WAIT]
specifies whether this logically recoverable intrapartition queue can wait for the resolution of an indoubt failure. Possible values are:

YES|NO

[WAIT_ACTION]
specifies the action to be taken if this logically recoverable intrapartition queue suffers an indoubt failure. Possible values are:

QUEUE|REJECT

[NUM_ITEMS] states the number of committed items in the queue.

[OPEN_STATUS]
indicates whether the queue is open. Possible values are:

OPEN|CLOSED

[RECORD_FORMAT]
specifies the format of the records held on the extrapartition queue. Possible values are:

FIXUNB|FIXUNBA|FIXUNBM|FIXBLK|FIXBLKA|FIXBLKM|
VARBLK|VARBLKA|VARBLKM|VARUNB|VARUNBA|
VARUNBM|UNDEFINED

[RECORD_LENGTH]
specifies the record length of the extrapartition queue.

[RECOVERY] specifies the recovery type of an intrapartition queue. Possible values are:

NO|PH|LG

[REMOTE_NAME]
specifies the remote name of the queue if this is a remote queue definition.

[REWIND] specifies where the tape is positioned in relation to the end of the data set. This input parameter applies to extrapartition queues only. Possible values are:

REREAD|LEAVE

[TD_QUEUE_TOKEN]
states which token is associated with this queue.

[TD_TYPE] specifies the queue type. Possible values are:

EXTRA|INTRA|INDIRECT|REMOTE

[TRIGGER_LEVEL]
specifies the trigger level of the intrapartition queue.

[TYPE_FILE] specifies whether this queue is:
- An input queue
- An output queue
- Whether it is a queue that is to be read backwards.

Possible values are:

INPUT|OUTPUT|RDBACK

[USERID_TOKEN]
indicates which token is associated with the USERID that was specified for this intrapartition queue.

[SYSOUTCLASS]
specifies the SYSOUT class to be used for the associated output extrapartition queue.

[BLOCK_LENGTH]
specifies the block length of an extrapartition queue.

RESPONSE is Transient Data's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND DCT_ERROR DIRECTORY_MGR_ERROR LOGIC_ERROR
EXCEPTION	QUEUE_NOT_FOUND

TDTM gate, START_BROWSE_TDQDEF function

The START_BROWSE_TDQDEF function of the TDTM gate initiates a browse from a specified queue, or from the start of the DCT.

Input parameters

START_AT specifies a queue from which the browse should start.

Output parameters

BROWSE_TOKEN is returned and uniquely identifies this browse session.

RESPONSE is Transient Data's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] returned when RESPONSE is DISASTER. Possible values are: ABEND, DCT_ERROR, DIRECTORY_MGR_ERROR, and LOGIC_ERROR

TDTM gate, GET_NEXT_TDQDEF function

The GET_NEXT_TDQDEF function of the TDTM gate returns information about a queue as part of a browse operation.

Input parameters

BROWSE_TOKEN identifies the browse session.

Output parameters

QUEUE_NAME is the name of the queue.

[ATI_FACILITY]
specifies the facility associated with this intrapartition queue when a trigger transaction is attached. Possible values are:

TERMINAL|FILE|SYSTEM

[ATI_TERMID] specified together with the FACILITY option, FACILITY_ID identifies the facility that the trigger transaction should be associated with.

[ATI_TRANID] specifies the ATI transaction to be invoked when the trigger level is reached.

[BUFFER_NUMBER]
specifies the number of buffers to be associated with an extrapartition queue.

[DDNAME] specifies the DDNAME to be associated with an extrapartition queue.

[DISPOSITION]
specifies the disposition of the data set to be associated with an extrapartition queue. Possible values are:

SHR|OLD|MOD

[DSNAME] specifies the DSNAME of the data set to be associated with the extrapartition queue.

[EMPTY_STATUS]
indicates whether the queue contains any records, and whether the queue is full. This option applies to extrapartition queues only. Possible values are:

FULL|EMPTY|NOTEMPTY

[ENABLE_STATUS]
indicates the status of the queue. Possible values are:

ENABLED|DISABLING|DISABLED

[ERROR_OPTION]
specifies what action is to be taken in the event of an I/O error. This option applies to extrapartition queues only. Possible values are:

IGNORE|SKIP

[INDIRECT_DEST]
specifies the destination queue if this queue is an indirect queue.

[WAIT] specifies whether this logically recoverable intrapartition queue can wait for the resolution of an indoubt failure. Possible values are:

YES|NO

[WAIT_ACTION]
specifies the action to be taken if this logically recoverable intrapartition queue suffers an indoubt failure. Possible values are:

QUEUE|REJECT

[NUM_ITEMS] states the number of committed items in the queue.

[OPEN_STATUS]
indicates whether the queue is open. Possible values are:

OPEN|CLOSED

[RECORD_FORMAT]
specifies the format of the records held on the extrapartition queue. Possible values are:

FIXUNB|FIXUNBA|FIXUNBM|FIXBLK|FIXBLKA|FIXBLKM|
VARBLK|VARBLKA|VARBLKM|VARUNB|VARUNBA|
VARUNBM|UNDEFINED

[RECORD_LENGTH]
specifies the record length of the extrapartition queue.

[RECOVERY]
specifies the recovery type of an intrapartition queue. Possible values are:

NO|PH|LG

[REMOTE_NAME]
specifies the remote name of the queue if this is a remote queue definition.

[REWIND]
specifies where the tape is positioned in relation to the end of the data set. This input parameter applies to extrapartition queues only. Possible values are:

REREAD|LEAVE

[TD_QUEUE_TOKEN]
states which token is associated with this queue.

[TD_TYPE]
specifies the queue type. Possible values are:

EXTRA|INTRA|INDIRECT|REMOTE

[TRIGGER_LEVEL]
specifies the trigger level of the intrapartition queue.

[TYPE_FILE]
specifies whether this queue is:
- An input queue
- An output queue
- Whether it is a queue that is to be read backwards.

Possible values are:

INPUT|OUTPUT|RDBACK

[USERID_TOKEN]
indicates which token is associated with the USERID that was specified for this intrapartition queue.

[SYSOUTCLASS]
specifies the SYSOUT class to be used for the associated output extrapartition queue.

[BLOCK_LENGTH]
specifies the block length of an extrapartition queue.

RESPONSE
is Transient Data's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND DCT_ERROR DIRECTORY_MGR_ERROR LOGIC_ERROR
EXCEPTION	NO_MORE_DATA_AVAILABLE
INVALID	INVALID_BROWSE_TOKEN

TDTM gate, END_BROWSE_TDQDEF function

The END_BROWSE_TDQDEF function of the TDTM gate terminates a browse session.

Input parameters

BROWSE_TOKEN
identifies the browse session.

Output parameters

RESPONSE is Transient Data's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] returned when RESPONSE is DISASTER, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND DCT_ERROR DIRECTORY_MGR_ERROR LOGIC_ERROR
INVALID	INVALID_BROWSE_TOKEN

TDTM gate, SET_TDQUEUE function

The SET_TDQUEUE function of the TDTM gate updates attributes of an installed transient data queue.

Input parameters

QUEUE_NAME identifies the queue to be updated.

[ATI_FACILITY]
specifies the type of facility associated with this queue. Possible values are:

TERMINAL|FILE|SYSTEM

[ATI_TERMID] indicates whether the ATI facility is to be updated.

[ATI_TRANID] indicates whether the ATI transaction is to be updated.

[ATI_USERID] indicates whether the USERID associated with the ATI transaction is to be updated.

[USERID_TOKEN]
is the token that is supplied by the user domain when the userid is added to the system.

Output parameters

OLD_USER_TOKEN
identifies the token associated with a previous USERID.

RESPONSE is Transient Data's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND DCT_ERROR DIRECTORY_MGR_ERROR LOGIC_ERROR CATALOG_WRITE_ERROR
EXCEPTION	IS_CXRF NOT_CLOSED DISABLE_PENDING NOT_DISABLED QUEUE_IS_INDOUBT QUEUE_NOT_FOUND

TDTM gate, DISCARD_TDQDEF function

The DISCARD_TDQDEF function of the TDTM gate deletes an installed transient data queue definition and removes it from the catalog. A DELETEQ command is issued as part of the discard process.

Input parameters

QUEUE_NAME identifies the queue to be discarded.

[TD_QUEUE_TOKEN]
can be specified instead of QUEUE_NAME. TD_QUEUE_TOKEN identifies the queue to be discarded.

Output parameters

RESPONSE
is Transient Data's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DCT_ERROR DIRECTORY_MGR_ERROR LOGIC_ERROR CATALOG_DELETE_FAILED
EXCEPTION	NAME_STARTS_WITH_C NOT_CLOSED NOT_DISABLED DISABLE_PENDING QUEUE_NOT_FOUND

TDTM gate, COMMIT_TDQDEFS function

The COMMIT_TDQDEFS function of the TDTM gate catalogs all installed transient data queue definitions as part of cold start processing.

Input parameters

TOKEN
specifies the catalog to which the queue definitions are to be written.

Output parameters

RESPONSE
is Transient Data's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
returned when RESPONSE is DISASTER. Possible values are: DIRECTORY_MGR_ERROR, CATALOG_WRITE_FAILED, and ABEND.

TDXM gate, BIND_FACILITY function

The BIND_FACILITY function of the TDXM gate is used to associate a transaction with the definition for the transient data queue that caused the transaction to be trigger-level attached, where the principal facility is the queue itself (that is there is no terminal associated with the queue).

Input parameters

None.

Output parameters

RESPONSE
is Transient Data's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

REASON
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

TDXM gate, BIND_SECONDARY_FACILITY function

The BIND_SECONDARY_FACILITY function of the TDXM gate is used to associate a transaction with the definition for a transient data queue that has caused the transaction to be trigger-level attached (where the principal facility is a terminal and the secondary facility is the transient data queue itself).

Input parameters

None.

Output parameters

FACILITY_NAME
is the name of the transient data queue. The queue is the secondary facility and has been associated with this transaction.

RESPONSE
is Transient Data's response to the call. It can have any of the following values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

REASON
is returned when RESPONSE is DISASTER. Possible value is ABEND.

TDXM gate, RELEASE_FACILITY function

The RELEASE_FACILITY function of the TDXM gate is used to disassociate a transaction from the TD queue. (The principal facility type is either TERMINAL or TDQUEUE.)

Input parameters

TERMINATION_TYPE
is the type of transaction termination. It can have either of these values:

`NORMAL ABNORMAL`

[RESTART_REQUESTED]
indicates whether or not the transaction is to be restarted. It can have either of these values:

`YES|NO`

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

REASON
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	RESTART_FAILURE

TDXM gate, INQUIRE_FACILITY function

The INQUIRE_FACILITY function of the TDXM gate is used to inquire about the transient data facilities that support facility manager calls from the transaction manager domain.

Input parameters

[FACILITY_TOKEN]
is the token identifying the transaction that has been trigger-level attached.

Output parameters

FACILITY_NAME
is the four-character name of the transaction that has been trigger-level attached.

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

TFAL gate, ALLOCATE function

The ALLOCATE function of the TFAL gate is used to allocate a terminal for a transaction.

Input parameters

REQUEST_ID
is the four-character transaction identifier initiating the attach.

[MODE_NAME]
is the eight-character mode-name of the terminal to be attached.

SYSTEM_TOKEN
is the token identifying the CICS region to which the terminal is to be attached.

[PRIVILEGED]
indicates whether or not the terminal is to be attached as a privileged terminal. It can have either of these values:

`YES|NO`

[NON_PURGEABLE]
indicates whether or not the terminal is to be purgeable. It can have either of these values:

YES|NO

Output parameters

TERMINAL_TOKEN
is the token identifying the terminal that has been attached.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	GETMAIN_FAILED LOGIC_ERROR
EXCEPTION	ALLOCATE_FAILURE ALLOCATE_PURGED
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, CANCEL_AID function

The CANCEL_AID function of the TFAL gate is used to cancel a terminal-transaction AID.

Input parameters

TERMID
is the four-character terminal identifier.

TRANID
is the four-character transaction identifier.

TERM_OWNER_NETNAME
is the APPLID of the CICS region that "owns" the terminal.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, CANCEL_AIDS_FOR_CONNECTION function

The CANCEL_AIDS_FOR_CONNECTION function of the TFAL gate is used to cancel AIDs for the given CICS region.

Input parameters

SYSTEM_TOKEN
is the token identifying the CICS region.

CALLER
is the method used to call this function. It can have either of these values:

BUILDER|API

FORCE
indicates whether or not system AIDs are to be canceled. It can have either of these values:

YES|NO

FACILITY
indicates the facility type associated with the AIDs. It can have either of these values:

CONNECTION|TERMINAL

Output parameters

[AIDS_CANCELLED]
indicates whether or not AIDs were canceled as a result of this request. It can have either of these values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NULL_SYSTEM_TOKEN
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, CANCEL_AIDS_FOR_TERMINAL function

The CANCEL_AIDS_FOR_TERMINAL function of the TFAL gate is used to cancel all AIDs for the given terminal.

Input parameters

Note: Specify either TERMID or TERMINAL_TOKEN, not both.

TERMID is the four-character terminal identifier.

TERMINAL_TOKEN
is the token identifying the terminal.

CALLER is the method used to call this function. It can have one of these values:

BUILDER|API|BUILDER_REMDEL

FORCE indicates whether or not system AIDs are to be canceled. It can have either of these values:

YES|NO

FACILITY indicates the facility type associated with the AIDs. It can have either of these values:

CONNECTION|TERMINAL

Output parameters

[AIDS_CANCELLED]
indicates whether or not AIDs were canceled as a result of this request. It can have either of these values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NULL_TERMINAL_TOKEN,
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, CHECK_TRANID_IN_USE function

The CHECK_TRANID_IN_USE function of the TFAL gate is used to check whether any of the AID chains contain ferrences to the given TRANID

Input parameters

TRANID is the four-character transaction identifier.

Output parameters

IN_USE indicates whether or not the transaction identifier (specified by the TRANID parameter) is in use. It can have either of these values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, DISCARD_AIDS function

The DISCARD_AIDS function of the TFAL gate is used to attach a task which will release start data and free the AIDs in the chain addressed by the AID_TOKEN

Input parameters

AID_TOKEN is the token identifying the chain of AIDs.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, FIND_TRANSACTION_OWNER function

The FIND_TRANSACTION_OWNER function of the TFAL gate is used to determine the CICS region that owns the given transaction (that is, at which the transaction instance originated).

Input parameters

TERMINAL_TOKEN
is the token identifying the terminal.

TRANID is the four-character transaction identifier.

Output parameters

TRAN_OWNER_SYSID
is the four-character system identifier for the CICS region that owns the transaction instance.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND TOR_LINK_NOT_ACTIVE
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, GET_MESSAGE function

The GET_MESSAGE function of the TFAL gate is used to get a message from a terminal.

Input parameters

TERMINAL_TOKEN
is the token identifying the terminal.

PREVIOUS_AID_TOKEN
is the AID token identifying the previous transaction that ran at this terminal.

Output parameters

AID_TOKEN is the AID token identifying the current transaction for which the message was got.

TSQUEUE_NAME is the eight-character name of the temporary storage queue name of the message whose BMS AID was found.

BMS_TITLE_PRESENT
indicates whether or not a BMS title is present on the terminal. It can have either of these values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, INITIALIZE_AID_POINTERS function

The INITIALIZE_AID_POINTERS function of the TFAL gate is used to initialize the AID pointers for the given CICS region.

Input parameters

SYSTEM_TOKEN is the token identifying the CICS region.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, INQUIRE_ALLOCATE_AID function

The INQUIRE_ALLOCATE_AID function of the TFAL gate is used to inquire about the AIDs allocated for the given CICS region.

Input parameters

SYSTEM_TOKEN is the token identifying the CICS region.

[PRIVILEGED] indicates whether or not to inquire only about privileged ISC type AIDs. It can have either of these values:

YES|NO

Output parameters

EXISTS indicates whether or not the AID exists. It can have either of these values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, LOCATE_AID

The LOCATE_AID function of the TFAL gate is used for automatic transaction initiation to determine the AID for the specified terminal, and if found, to use the transaction identifier from the AID to attach the task.

Input parameters

TERMID is the four-character terminal-identifier.

[TYPE] denotes the type of AID to be located. It can have one of these values:

BMS|PUT|INT|TDP|ISC|REMDEL

Output parameters

[TRANID] is the four-character transaction identifier associated with the specified terminal.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, LOCATE_REMDEL_AID

The LOCATE_REMDEL_AID function of the TFAL gate is used to determine the AID (for a delete remote TERMINAL definition request) for the specified system (SYSTEM_TOKEN specified) or after the given (PREVIOUS_AID_TOKEN specified).

Input parameters

SYSTEM_TOKEN is the token identifying the CICS region.

PREVIOUS_AID_TOKEN
is the AID token identifying the previous transaction that ran at this terminal.

Output parameters

AID_TOKEN is the AID token identifying the transaction to be deleted.

TARGET_SYSID is the four-character system identifier for the target CICS system.

TERMID is the four-character terminal identifier from the REMDEL AID.

TERM_OWNER_NETNAME
is the eight-character netname from the REMDEL AID.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, LOCATE_SHIPPABLE_AID

The LOCATE_SHIPPABLE_AID function of the TFAL gate is used to determine an AID (for a delete remote TERMINAL definition request or for a remote terminal request) to be shipped to the specified system.

Input parameters

SYSTEM_TOKEN is the token identifying the CICS region.

Output parameters

AID_TOKEN is the AID token identifying the transaction to be deleted.

LAST Indicates that either:
- there is a single qualifying AID or all qualifying AIDs have the same AIDTRMID (YES), or
- in addition to the AID returned there are other qualifying AIDs (NO)

It can have either of these values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, MATCH_TASK_TO_AID function

The MATCH_TASK_TO_AID function of the TFAL gate is used to inquire about AIDs for the given terminal and transaction.

Input parameters

TERMINAL_TOKEN
is the token identifying the terminal.

TRANID is the four-character transaction identifier.

Output parameters

TDQUEUE_NAME is the eight-character name of the transient data queue for the AID.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND MATCHED_TERMID_ONLY
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, PURGE_ALLOCATE_AIDS

The PURGE_ALLOCATE_AIDS function of the TFAL gate is used to delete purgeable allocate AIDs for a given connection after user exit XZIQUE in DFHZISP has issued return code 8 (delete all) or return code 12 (delete all for given modegroup).

Input parameters

SYSTEM_TOKEN is the token identifying the CICS region.

[MODE_NAME] The name of the modegroup. If this parameter is omitted, the default is all modegroups.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, RECOVER_START_DATA

The RECOVER_START_DATA function of the TFAL gate is used to retrieve a PUT-type AID stored in a DWE and rechain it onto the TCTSE in front of the first AID for the terminal.

Input parameters

AID_TOKEN is the AID token identifying the transaction to be deleted.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	NULL_SYSTEM_TOKEN GETMAIN_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, REMOTE_DELETE

The REMOTE_DELETE function of the TFAL gate is used to chain a REMOTE DELETE (REMDEL) AID onto the system entry of the specified target CICS region. The REMDEL AID tells the target region to delete its shipped definition of the specified terminal.

Input parameters

TARGET_SYSID is the four-character system identifier for the target CICS region.

TERMINAL_TOKEN
is the token identifying the terminal.

TERMID is the four-character terminal identifier for the terminal associated with the transaction.

TERM_OWNER_NETNAME
Is the VTAM APPLID of the CICS region that "owns" the terminal.

Note: The terminal identifier can either be specified as TERMID and TERM_OWNER_NETNAME (where TERMID is the name known in the terminal owning system), or it can be specified by TERMINAL_TOKEN if the TCTTE address is known.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	GETMAIN_FAILED
EXCEPTION	TOR_LINK_NOT_ACTIVE
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, REMOVE_EXPIRED_AID

The REMOVE_EXPIRED_AID function of the TFAL gate is used to search all AID chains for a BMS AID that has yet to be initiated and which matches the eligibility parameters. Unchain the first such AID found, copy details from the AID into the caller's parameter list, and freemain the AID.

Input parameters

[NORMAL_EXPIRY_TIME]
is the normal threshold time.

[ADJUSTED_EXPIRY_TIME]
is the adjusted threshold time.

[MSGID] is the BMS message identifier

[LDC] is the logical device code

Note: If MSGID and LDC are specified, the expiry time is not checked.

Output parameters

TSQUEUE_NAME is the eight-character name of the temporary storage queue name of the message whose BMS AID was found.

TRANID is the four-character transaction identifier associated with the specified terminal.

TERMID is the four-character terminal identifier for the terminal associated with the transaction.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, REMOVE_EXPIRED_REMOTE_AID

The REMOVE_EXPIRED_REMOTE_AID function of the TFAL gate is used to search for an uninitiated remote AID which is older than the expiry time specified by the caller, unchain the AID, and cleanup any associated resources.

Input parameters

NORMAL_EXPIRY_TIME
is the normal threshold time.

ADJUSTED_EXPIRY_TIME
is the adjusted threshold time.

Output parameters

TRANID is the four-character transaction identifier associated with the specified terminal.

TERMID is the four-character terminal identifier for the terminal associated with the transaction.

TERM_OWNER_SYSID
is the system identifier of the CICS region that "owns" the terminal.

SHIPPED identifies whether the AID has been shipped. It can have either of these values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, REMOVE_MESSAGE

The REMOVE_MESSAGE function of the TFAL gate is used to:

1. Find an uninitiated BMS AID for the specified terminal
2. Unchain and freemain the AID, provided that the AID security fields match those of the currently signed-on operator
3. Return the TS queue name from the AID.

Input parameters

TERMINAL_TOKEN
is the token identifying the terminal.

[MSGID] is the BMS message identifier

Output parameters

TSQUEUE_NAME is the eight-character name of the temporary storage queue name for the message whose BMS AID was found.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND SECURITY_MISMATCH
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, REMOVE_REMOTE_DELETES

The REMOVE_REMOTE_DELETES function of the TFAL gate is used to unchain and freemain all REMDEL AIDs from the AID chain of the specified system entry. Optional parameters TERMID and TERM_OWNER_NETNAME may be specified; in which case only those REMDEL AIDs which match the specified values are removed.

Input parameters

TARGET_SYSID is the four-character system identifier for the target CICS region.

SYSTEM_TOKEN is the token identifying the CICS region.

Note: Specify either the TARGET_SYSID parameter or the SYSTEM_TOKEN parameter, not both.

[TERMID] is the four-character terminal identifier for the terminal associated with the transaction.

[TERM_OWNER_NETNAME]
is the netname of the region that "owns" the terminal.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, REROUTE_SHIPPABLE_AIDS

The REROUTE_SHIPPABLE_AIDS function of the TFAL gate is used to redirect AIDs for remote terminals from one remote system to another.

Input parameters

ORIGINAL_SYSTEM_TOKEN
is the token identifying the remote system which was the AIDs' original target.

TARGET_SYSTEM_TOKEN
is the token identifying the remote system which is the AIDs' new target.

TERMINAL_NETNAME
is the eight-character NETNAME which identifies the terminal whose AIDs are to be rerouted.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, RESCHEDULE_BMS

The RESCHEDULE_BMS function of the TFAL gate is used to build a BMS AID and chain it to the front of the AID queue.

Input parameters

TERMINAL_TOKEN
is the token identifying the terminal.

TRANID
is the four-character transaction identifier associated with the specified terminal.

TSQUEUE_NAME
is the eight-character name of the temporary storage queue name of the message whose BMS AID was found.

BMS_TIMESTAMP
Timestamp for BMS AID. Used to test if AID is older than specified EXPIRY_TIME.

[OPIDENT]
Identifies the operator

Note: You can specify either the OPIDENT parameter or the OPCLASS parameter, not both.

[OPCLASS]
Identifies the operator class.

Note: You can specify either the OPIDENT parameter or the OPCLASS parameter, not both.

[BMS_TITLE_PRESENT]
Indicates if title in message control record. You can specify either of these values:
YES|NO

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	GETMAIN_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, RESET_AID_QUEUE

The RESET_AID_QUEUE function of the TFAL gate is used to:

1. Give ALP a chance to reset the AID queue when a transaction ends
2. Give ALP a chance to bid for the use of the terminal if ATI tasks are waiting.

Input parameters

TERMINAL_TOKEN
is the token identifying the terminal.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, RESTORE_FROM_KEYPOINT

The RESTORE_FROM_KEYPOINT function of the TFAL gate is used to: reschedule a chain of AIDs that we restored from the catalog during CICS system initialization.

Input parameters

AID_TOKEN A token denoting the chain of AIDs which are to be rescheduled.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, RETRIEVE_START_DATA

The RETRIEVE_START_DATA function of the TFAL gate is used to return the AID address and temporary storage queue name associated with the start data for the specified transaction and terminal.

Input parameters

TERMINAL_TOKEN
is the token identifying the terminal.

TRANID is the four-character transaction identifier associated with the specified terminal.

Output parameters

TSQUEUE_NAME is the eight-character name of the temporary storage queue name of the message whose BMS AID was found.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, SCHEDULE_BMS

The SCHEDULE_BMS function of the TFAL gate is used to: schedule a BMS AID.

Input parameters

TERMID is the four-character terminal identifier for the terminal associated with the transaction.

TRANID is the four-character transaction identifier associated with the specified terminal.

TSQUEUE_NAME is the eight-character name of the temporary storage queue name of the message whose BMS AID was found.

BMS_TIMESTAMP
is the timestamp for the BMS AID. This is used to test if the AID is older than its EXPIRY_TIME.

[OPIDENT] Identifies the operator.

Note: You can specify either the OPIDENT parameter or the OPCLASS parameter, not both.

[OPCLASS] Identifies the operator class.

Note: You can specify either the OPIDENT parameter or the OPCLASS parameter, not both.

[BMS_TITLE_PRESENT]
Indicates if the title is in the message control record. You can specify either of these values:

YES|NO

[TERMINAL_NETNAME]
is the eight-character NETNAME which identifies the terminal whose AIDs are to be rerouted.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	GETMAIN_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, SCHEDULE_START

The SCHEDULE_START function of the TFAL gate is used to schedule a PUT or INT type AID

Input parameters

TRANID is the four-character transaction identifier associated with the specified terminal.

TERMID is the four-character terminal identifier for the terminal associated with the transaction.

[TRAN_OWNER_SYSID]
is the system identifier of the CICS region that "owns" the transaction.

[TERM_OWNER_SYSID]
is the system identifier of the CICS region to which the request should be shipped.

Note: You can specify either the TERM_OWNER_SYSID parameter or TERM_OWNER_NETNAME parameter, not both.

[TERM_OWNER_NETNAME]
is the system identifier of the CICS region to which the request should be shipped.

Note: You can specify either the TERM_OWNER_SYSID parameter or TERM_OWNER_NETNAME parameter, not both.

[ROUTED_FROM_TERMID]
is the four-character terminal identifier for the terminal from which a task was transaction-routed to issue this START request.

[SHIPPED_VIA_SESSID]
is the identifier of the session via which this START request was function shipped.

[MODE_NAME] is the mode name to be used

[TSQUEUE_NAME]
is the name of the temporary storage queue which contains the data associated with the START request.

[FEPI] indicates that this is a FEPI START request. It can have either of these values:

YES|NO

[RECOVERABLE_DATA]
indicates that the request is associated with recoverable data It can have either of these values:

YES|NO

[IN_DOUBT] indicates that the Unit of Work making the request is in doubt, and the request should not be scheduled until the Unit of Work is committed. It can have either of these values:

YES|NO

[TERMINAL_NETNAME]
is the eight-character NETNAME of the terminal associated with the transaction.

[SHIPPED_VIA_SYSID]
identifies the connection via which this request was function shipped or transaction routed.

[TOR_NETNAME]
is the netname of the CICS region that owns the terminal.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	GETMAIN_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, SCHEDULE_TDP

The SCHEDULE_TDP function of the TFAL gate is used to schedule a TDP type AID.

Input parameters

TRANID is the four-character transaction identifier associated with the specified terminal.

TERMID is the four-character terminal identifier for the terminal associated with the transaction.

TDQUEUE_NAME is the destination identifier for the TD queue.

[TERMINAL_NETNAME]
is the eight-character NETNAME of the terminal associated with the transaction.

Output parameters

AID_TOKEN is the AID token identifying the transaction to be deleted.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	GETMAIN_FAILED
EXCEPTION	UNKNOWN_TRANID
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, SLOWDOWN_PURGE

The SLOWDOWN_PURGE function of the TFAL gate is used to:

1. Search the specified system entry's AID chain for the first allocate-type AID associated with a stall-purgeable task
2. Cancel the identified transaction.

Input parameters

SYSTEM_TOKEN is the four-character terminal identifier for the terminal associated with the transaction.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, TAKE_KEYPOINT

The TAKE_KEYPOINT function of the TFAL gate is used to return a chain of AIDs which are to be written to the global catalog.

Input parameters

None.

Output parameters

AID_TOKEN is the token identifying the chain of AIDs.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, TERM_AVAILABLE_FOR_QUEUE

The TERM_AVAILABLE_FOR_QUEUE function of the TFAL gate is used, when a terminal becomes available for allocation, to give DFHALP the chance to attach or resume a task which requires this terminal.

Input parameters

TERMINAL_TOKEN
is the token identifying the terminal.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	GETMAIN_FAILED ATTACH_ERROR
EXCEPTION	NOT_FOUND
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, TERMINAL_NOW_UNAVAILABLE

The TERMINAL_NOW_UNAVAILABLE function of the TFAL gate is used to perform required actions when a terminal or connection becomes unavailable.

Input parameters

TERMINAL_TOKEN
is the token identifying the terminal.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, UNCHAIN_AID

The UNCHAIN_AID function of the TFAL gate is used to unchain and optionally freemain the specified AID.

Input parameters

AID_TOKEN is the AID token identifying the transaction to be deleted.

FREEMAIN indicates whether freemain is wanted. It can have either of these values:

YES|NO

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFAL gate, UPDATE_TRANNUM_FOR_RESTART

The UPDATE_TRANNUM_FOR_RESTART function of the TFAL gate is used to update the AID's TRANNUM to that of the restarted task.

Input parameters

TERMINAL_TOKEN
is the token identifying the terminal.

ORIGINAL_TRANNUM
is the TRANNUM set in the AID when original task was attached.

NEW_TRANNUM
is the new TRANNUM to be set in the AID.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NULL_TERMINAL_TOKEN
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFBF gate, BIND_FACILITY function

The BIND_FACILITY function of the TFBF gate is used to associate a transaction with the terminal.

Input parameters

[PROFILE]
is the eight-character name of the profile to be used to associate the transaction and terminal.

[PARTITIONSET_NAME]
is the eight-character name of a partition set. This parameter is used only of the value of PARTITIONSET is NAME.

[PARTITIONSET]
indicates if a partition set is to be used for the terminal facility. It can have any of these values:

NONE|NAME|OWN|KEEP

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND REMOTE_SCHEDULE_FAILURE SECURITY_FAILURE TABLE_MANAGER_FAILURE
EXCEPTION	NO_TERMINAL TRANSACTION_ABEND
INVALID	INVALID_FORMAT INVALID_FUNCTION

TFIQ gate, INQUIRE_TERMINAL_FACILITY function

The INQUIRE_TERMINAL_FACILITY function of the TFIQ gate is used to inquire about attributes of a named terminal facility.

Input parameters

Note: Specify a value for either the TRANSACTION_TOKEN or TERMINAL_TOKEN parameter, not both.

[TRANSACTION_TOKEN]
is a token identifying a transaction for which you want to inquire about the associated terminal.

[TERMINAL_TOKEN]
is a token identifying a terminal.

Output parameters

[FACILITY_NAME]
is the four-character name of the terminal facility.

[NETNAME]
is the eight-character netname of the terminal facility.

[PSEUDO_CONV_COMMAREA]
is a block into which the communications area for a pseudo-conversational transaction is copied.

[TERMINAL_TRAFFIC_READ]
indicates whether or not reading is supported. It can have either of these values:

YES|NO

[TERMINAL_TRAFFIC_WRITE]
indicates whether or not writing is supported. It can have either of these values:

YES|NO

[TERMINAL_USER_AREA]
is a block into which the terminal user area is copied.

[NATIONAL_LANGUAGE_IN_USE]
is the three-character code indicating the national language in use for the terminal facility. (See Table 120 on page 1318.)

[INSPECT_DATA]
is a token indicating the Language Environment runtime options for the terminal facility.

[STORAGE_FREEZE]
indicates whether or not storage normally freed during the processing of a transaction for the terminal facility is to be frozen. (The frozen storage is not freed until the end of the transaction.) It can have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_TERMINAL
INVALID	INVALID_TERMINAL_TYPE

TFIQ gate, SET_TERMINAL_FACILITY function

The SET_TERMINAL_FACILITY function of the TFIQ gate is used to set attributes of a named terminal facility.

Input parameters

Note: Specify a value for either the TRANSACTION_TOKEN or TERMINAL_TOKEN parameter, not both.

[TRANSACTION_TOKEN]
is a token identifying a transaction for which you want to inquire about the associated terminal.

[TERMINAL_TOKEN]
is a token identifying a terminal.

[COUNT_STORAGE_VIOLATION]
indicates whether or not storage violations are to be counted for this terminal facility. It can have either of these values:

YES|NO

[INPUTMSG]
is a block into which the input message for a pseudo-conversational transaction is copied.

[PSEUDO_CONV_NEXT_TRANSID]
is the four-character identifier of the transaction to which control is passed on a normal return from a pseudo-conversational transaction (to which the pseudo_conversational data is passed).

[PSEUDO_CONV_COMMAREA]
is a block into which the communications area for a pseudo-conversational transaction is copied.

[PSEUDO_CONV_IMMEDIATE]
is the four-character identifier of the transaction to which control is passed on an immediate return from a pseudo-conversational transaction (to which the pseudo_conversational data is passed).

[NATIONAL_LANGUAGE_IN_USE]
is the three-character code indicating the national language in use for the terminal facility. (See Table 120 on page 1318.)

[INSPECT_DATA]
is a token indicating the Language Environment runtime options for the terminal facility.

[STORAGE_FREEZE]
indicates whether or not storage normally freed during the processing of a transaction for the terminal facility is to be frozen. (The frozen storage is not freed until the end of the transaction.) It can have either of these values:

YES|NO

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_TERMINAL PERMANENT_TRANSID
INVALID	INVALID_TERMINAL_TYPE

TFIQ gate, INQUIRE_MONITOR_DATA function

The INQUIRE_MONITOR_DATA function of the TFIQ gate is used to inquire about monitoring data of the terminal facility.

Input parameters

None.

Output parameters

[FACILITY_TYPE]
indicates the type of terminal facility. It can have any of these values:

LU61|LU62|IRC|IRC_XCF|OTHER

[FACILITY_NAME]
is the four-character name of the terminal facility.

[NETNAME] is the eight-character netname of the terminal facility.

[INPUT_MSG_LENGTH]
is the length (in bytes) of the input message for the terminal facility.

[SERVICE_REPORTING_CLASS]
is a token indicating the service reporting class for the terminal facility (for MVS workload manager purposes).

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_TERMINAL

Application domain's generic gates

Table 31 summarizes the application domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 31. Application domain's generic gates

Gate	Trace	Function	Format
APDM	AP 0900 AP 0901	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
APDS	AP 0500 AP 0501	TASK_REPLY PURGE_INHIBIT_QUERY	DSAT
APST	AP D400 AP D401	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
APSM	AP F110 AP F111	STORAGE_NOTIFY	SMNT
APTI	AP F300 AP F301	NOTIFY	TISR

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats, as follows:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669
Format DSAT—"Dispatcher domain's generic formats" on page 717
Format STST—"Statistics domain's generic format" on page 1198
Format SMNT—"Storage manager domain's generic formats" on page 1159
Format TISR—"Timer domain's generic format" on page 1203

Application domain's generic formats

Table 32 describes the generic formats owned by the application domain and shows the functions performed on the calls.

Table 32. Generic formats owned by application domain

Format	Calling module	Function
APUE	DFHUEM	SET_EXIT_STATUS

In the descriptions of the formats that follow, the “input” parameters are input not to the application domain, but to the domain being called by the application domain. Similarly, the “output” parameters are output by the domain that was called by the application domain, in response to the call.

APUE format, SET_EXIT_STATUS function

The SET_EXIT_STATUS function of the APUE format is used to set the exit status at a specified exit point.

Input parameters

EXIT_POINT
is the name of the exit to be enabled or disabled.

EXIT_STATUS (ACTIVE|INACTIVE
indicates whether the exit is to be made active or inactive.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION INVALID_EXIT_POINT

Control blocks

The main CICS control block in the AP domain is the common system area (CSA), which exists from CICS system initialization time until CICS is closed down. The CSA contains:
- Register save area
- Pointers to the CICS control modules
- Control information
- System constants
- Time-control storage
- Work area for statistics
- Task abnormal termination interface
- Pointers to CICS system tables.

The CSA has an extension area known as the CSA optional features list. The address of the optional features list is held in CSAOPFLA in the CSA, and also in TCACSOAD in the TCA.

See *CICS Data Areas* for a detailed description of these control blocks.

There is also a user-defined work area, called the common work area (CWA). The user can govern the length and storage key of the CWA by using the WRKAREA and CWAKEY system initialization parameters.

The CWA is available to any task while it has control of the system (that is, for operations performed between requests to CICS).

Modules

Module	Function
DFHAPDM	AP domain/domain manager gate service module. Handles the following calls made by the domain manager to the AP domain: INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHAPEX	AP domain user exit service module. This module handles INVOKE_USER_EXIT made by several domains to the AP domain.
DFHAPID	Handles the following requests: PROFILE QUERY_NETNAME
DFHAPIQ	AP domain task data inquire and set gate service module. Handles the following call to the AP domain: INQ_APPLICATION_DATA
DFHAPJC	AP domain/journal gate service module. This module handles WRITE_JOURNAL_DATA calls made by the user exits' XPI.
DFHAPPIJ	Provides Language Environment PIPI service routines for getstore and freestore, for use in JVMs running under CICS. This module invokes DFHSMVP through DFHSMVPI to perform execute and monitor the MVS getmains and freemains.
DFHAPSM	AP domain storage notify gate service module.
DFHAPST	AP domain functional gate for statistics. This module accepts a request for and then supervises the copying and resetting of statistics counters in the AP domain by calling the appropriate DFHSTxx modules to access the counters.
DFHAPTC	AP domain module DFHAPTC provides a common mechanism for issuing TC macro calls.It issues the following calls: CANCEL CLOSE EXTRACT_PROCESS ISSUE NOTIFY LISTEN OPEN RECEIVE SEND SET_SESSION
DFHAPTI	AP domain timer domain gate service module. This module handles NOTIFY calls made by the timer domain to the AP domain.
DFHAPTIM	CICS interval control midnight task. This module deals with NOTIFY requests from the timer domain.
DFHAPTIX	CICS expiry analysis task. This module deals with NOTIFY requests from the timer domain.
DFHAPXM	AP domain/transaction manager gate service module. Handles the following calls made by the transaction manager to the AP domain: TRANSACTION_INITIALIZATION RMI_START_OF_TASK TRANSACTION_TERMINATION
DFHICXM	AP domain/interval control principal facility management gate service module. Handles the following calls made by the transaction manager to the AP domain: INQUIRE_FACILITY
DFHSAIQ	AP domain system data inquire and set gate service module. Handles the following calls to the AP domain: INQUIRE_SYSTEM SET_SYSTEM
DFHSRP	Default system recovery program for the AP domain. It includes the ABAB functions. For more information about DFHSRP, see Chapter 50, "System recovery program," on page 375.
DFHTDXM	AP domain/transient data principal facility management gate service module. Handles the following calls made by the transaction manager to the AP domain: BIND_FACILITY BIND_SECONDARY_FACILITY RELEASE_FACILITY INQUIRE_FACILITY

Module	Function
DFHTFBF	AP domain/terminal facility manager bind facility gate service module. Handles the following call made by the terminal facility manager to the AP domain: BIND_FACILITY
DFHTFIQ	AP domain/terminal facility manager inquire and set gate service module. Handles the following calls made by the terminal facility manager to the AP domain: INQUIRE_TERMINAL_FACILITY INQUIRE_MONITOR_DATA SET_TERMINAL_FACILITY
DFHTFRF	AP domain/terminal facility manager release facility gate service module. Handles the following calls made by the terminal facility manager to the AP domain: RELEASE_FACILITY

Exits

Various global user exit points are provided for this domain, and these are described under the appropriate functions in the rest of this book.

Trace

Various trace point IDs are provided for this domain, and these are described under the appropriate functions in the rest of this book.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 71. AP domain initialization program

The AP domain initialization program is resident only long enough to start up the AP domain.

Modules

The main initialization program is DFHAPSIP. DFHAPSIP calls a series of modules DFHSIA1, DFHSIB1, ..., DFHSIJ1, which complete initialization. DFHAPSIP receives control from DFHAPDM. For further information about DFHAPDM, see page "Modules" on page 595.

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for this function:
- AP 0700 (DFHSII1 add gate), for which the trace level is Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 72. AP domain KC subcomponent

The AP domain KC subcomponent does the following:

- Provides an enqueue facility
- Manages profile definitions (making use of table manager program (see “The FEPI Resource Manager work queues” on page 266)).
- Converts some DFHKC macro calls into dispatcher domain calls and transaction manager domain calls.

Design overview

This section describes the macro calls supported by the AP domain KC subcomponent.

DFHKC macro calls

ATTACH.
This call is converted into a transaction manager domain XMAT ATTACH call to create an instance of the requested transaction. This request is only used to create CICS system transactions and may not be used to attach a user transaction.

DEQ.
DEQ is used to reduce the use count of a resource previously enqueued on by this transaction. If the use count reaches zero, the resource is freed for use by another transaction. The NQED DEQUEUE service of the NQ domain is used for this function.

ENQ.
The caller passes a resource name or address. The AP domain KC subcomponent issues an NQED ENQUEUE request to the NQ domain.

INITIALIZE.
INITIALIZE is used during CICS initialization to tell the AP domain KC subcomponent to build profile table entries in storage.

PROFBROWSE.
This is used to browse profile table (PFT) entries.

PROFLOC.
This finds the profile table (PFT) entry for the profile ID passed.

REPLACE.
This replaces an existing profile table entry by a new version.

RESUME.
This call is converted into a dispatcher domain DSSR RESUME call to resume the suspended task.

WAIT.
Wait calls are converted into the appropriate dispatcher domain call.

WAITINIT.
This is used once during initialization to wait for the completion of an earlier INITIALIZE call.

Control blocks

Static storage area (SSA).
The AP domain KC subcomponent uses an SSA as a permanent work area. Field SSAKCP in the static storage area address list (as defined by the DSECT DFHSSADS) points to the AP domain KC subcomponents static storage area. The address of the static storage area address list is held in field CSASSA in the CSA optional features list.

See the *CICS Data Areas* manual for a detailed description of these control blocks.

Modules

The following are link-edited together to form the DFHKCP module:

Module	Function
DFHKCP	This is a startup routine that passes control to either DFHXCP or DFHXCPC.
DFHXCP	Processes DFHKC ATTACH, RESUME, and WAIT macro calls to the transaction manager and dispatcher and handles the DFHKC PROFLOC AND PROFBROWSE (profile locate and profile browse) services.

Module	Function
DFHXCPC	Processes DFHKC DEQ and ENQ macro calls to the AP domain KC subcomponent Receives DFHKC INITIALIZE, REPLACE, WAITINIT, and DISCARD macro calls to the transaction manager and passes them on to DFHKCQ.
DFHKCQ	Processes DFHKC INITIALIZE, REPLACE, WAITINIT, and DISCARD macro calls to the AP domain KC subcomponent.
DFHKCSC	Provides chain scanning facilities for the DISCARD TRANSACTION command.

Exits

There are two globasl user exit points in DFHEKC: XNQEREQ and XNQEREQC. See the *CICS Customization Guide* for further information.

Trace

The following point ID is provided for the AP domain KC subcomponent

- AP F0xx, for which the trace levels are AP 1, AP 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Dumps

F007
DFHXCP was called to process a AP domain KC subcomponent request but did not recognize the function code in the TCA.

External interfaces

The AP domain KC subcomponent calls the following domains: DS, GC, KE, ME, MN, NQ, SM, TR and XM.

The AP domain KC subcomponent calls the following CICS AP domain function:

- Table manager

Statistics

No statistics are created by the AP domain KC subcomponent

Chapter 73. AP domain termination program

The AP domain (system) termination program (DFHSTP) provides for an orderly shutdown of CICS. When an PERFORM SHUTDOWN or PERFORM TAKEOVER command is used, either on the CEMT transaction or by an EXEC CICS command, the DFHEIPSH program invokes DFHSTP to handle it.

Design overview

Figure 107 shows the relationships between the components of AP domain termination.

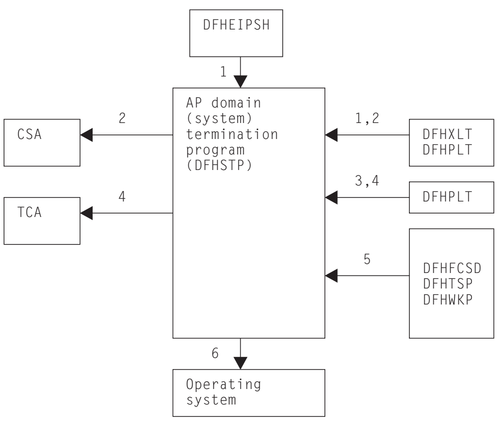


Figure 107. AP domain termination program interfaces

Notes:

1. When a PERFORM SHUTDOWN or PERFORM TAKEOVER command is used, either on the CEMT transaction or by an EXEC CICS command, the DFHEIPSH program:
   - Loads the transaction list table (XLT) and program list table (PLT) from the DFHRPL DD concatenation
   - Transfers control to DFHSTP by means of a DFHPGXE PREPARE_XCTL_EXEC domain call.

For an immediate shutdown, statistics are collected at the step described by 1. Following this, the resource managers and the subsystem interface are terminated; no load of tables, terminal quiescing, or execution of programs specified in the PLT occurs, that is to say the steps described in notes 601, 2, 3, and 4 are not performed on an immediate shutdown. Also, CICS files are not closed during step 5 on an immediate shutdown.

2. Terminal activity is quiesced via an indicator in the CSA. This tells terminal control not to attach any transactions other than those specified in the XLT and those specifying SHUTDOWN(ENABLED) in their associated TRANSACTION resource definitions. The termination task logically disconnects itself from the physical terminal to allow other activity on that terminal.
3. The termination task allows all other tasks (except any journal tasks) to complete before linking to the first program specified in the first portion of the PLT.
4. When all programs in the first portion of the PLT have executed, terminal activity is quiesced completely, using bit CSATQIM in CSASSI2 in the CSA. If monitoring is running, it is stopped. The ICE and AID chains are broken (addresses saved in the TWA), the IRC session is quiesced, and the programs specified in the second portion of the PLT are executed.
5. All open files managed by CICS file control are closed by the file control shutdown program, DFHFCSD; temporary-storage control, DFHTSP is requested to output its buffer; and a keypoint is taken by the warm keypoint program, DFHWKP.

6. Control is returned to the operating system, with or without a dump (depending upon the parameters specified in the shutdown request causing termination).

   For the high-performance option (HPO), the service request block (SRB) in the system queue area (SQA) is freed by using a CICS SVC (DFHCSVC).

Modules

DFHSTP

Exits

There is one global user exit point in DFHSTP: XSTERM. See the *CICS Customization Guide* for further information.

Trace

No trace points are provided for this function.

Chapter 74. Business Application Manager domain (BAM)

The business application manager domain (also sometimes known simply as "buisness application manager") is responsible for managing CICS business transaction services' (BTS) processes, process types and activities. It deals with the hardening of the associated data to BTS repository files. Along with scheduler services domain and event manager domain it forms the CICS BTS function.

Business application manager domain's specific gate

Table 33 summarizes the business application manager domain's specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 33. Business application manager domain's specific gate

Gate	Trace	Function	XPI
BATT	BA 0160 BA 0161	ADD_REPLACE_PROCSSTYPE INQUIRE_PROCESSTYPE START_BROWSE_PROCESSTYPE GET_NEXT_PROCESSTYPE END_BROWSE_PROCESSTYPE SET_PROCESSTYPE DISCARD_PROCESSTYPE COMMIT_PROCESSTYPE_TABLE	NO NO NO NO NO NO NO NO
BAXM	BA 0170 BA 0171	INIT_ACTIVITY_REQUEST BIND_ACTIVITY_REQUEST	NO NO
BAPR	BA 0110 BA 0111	ADD_PROCESS RUN_PROCESS LINK_PROCESS ACQUIRE_PROCESS CANCEL_PROCESS SUSPEND_PROCESS RESUME_PROCESS CHECK_PROCESS RESET_PROCESS	NO NO NO NO NO NO NO NO NO
BAAC	BA 0120 BA 0121	ADD_ACTIVITY RUN_ACTIVITY CHECK_ACTIVITY RETURN_END_ACTIVITY DELETE_ACTIVITY SUSPEND_ACTIVITY RESUME_ACTIVITY CANCEL_ACTIVITY LINK_ACTIVITY ACQUIRE_ACTIVITY RESET_ACTIVITY ADD_TIMER_REQUEST ADD_REATTACH_ACQUIRED	NO NO NO NO NO NO NO NO NO NO NO NO NO
BABR	BA 0150 BA 0151	STARTBR_ACTIVITY GETNEXT_ACTIVITY ENDBR_ACTIVITY INQUIRE_ACTIVITY STARTBR_CONTAINER GETNEXT_CONTAINER ENDBR_CONTAINER INQUIRE_CONTAINER STARTBR_PROCESS GETNEXT_PROCESS ENDBR_PROCESS INQUIRE_PROCESS INQUIRE_ACTIVATION COMMIT_BROWSE	NO NO NO NO NO NO NO NO NO NO NO NO NO NO
BACR	BA 0130 BA 0131	DELETE_CONTAINER GET_CONTAINER_INTO GET_CONTAINER_SET PUT_CONTAINER GET_CONTAINER_LENGTH	NO NO NO NO NO

Table 33. Business application manager domain's specific gate (continued)

Gate	Trace	Function	XPI
BACM	BA 01B0 BA 01B1	MOVE_CONTAINER	NO
BAGD	BA 0401 BA 0402	INQUIRE_DATA_LENGTH GET_DATA DESTROY_TOKEN ADDRESS_DATA RELEASE_DATA	NO NO NO NO

BATT gate, ADD_REPLACE_PROCESSTYPE function

The ADD_REPLACE_PROCESSTYPE function of the BATT gate is used to add a new process type definition or replace an existing process type definition. Process types are defined using RDO.

Input parameters

PROCESSTYPE_NAME
is an 8-character name.

FILE_NAME
is an 8-character name of the repository file to be associated with this process type. The file is defined using RDO.

AUDITLOG_NAME
is an 8-character name of the audit log to be associated with this process type. The log is defined using RDO.

AUDITLEVEL
determines the level of auditing to be undertaken for this process type. It can take the values:

OFF|PROCESS|ACTIVITY|FULL

USERRECORDS
indicates whether user audit records are to be written to the log. It can take the values:

YES|NO

CATALOG_PTDEF
indicates whether the definition should be written to the global catalog. It can take the values:

YES|NO

STATUS
indicates whether the process type definition should be installed in a disabled or enabled state. It can take the values:

DISABLED|ENABLED

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_DISABLED INSUFFICIENT_STORAGE

BATT gate, INQUIRE_PROCESSTYPE function

The INQUIRE_PROCESSTYPE function of the BATT gate is used to return information on the named process type.

Input parameters

PROCESSTYPE_NAME
is the 8-character name of the process type to be inquired upon.

Output parameters

FILE_NAME
is the 8-character name of the repository file associated with this process type.

AUDITLOG_NAME
is an 8-character name of the audit log associated with this process type.

AUDITLEVEL
identifies the level of auditing for this process type. It can take the values:

OFF|PROCESS|ACTIVITY|FULL

USERRECORDS
indicates whether user audit records are to being written to the log. It can take the values:

YES|NO

STATUS
indicates the status of the process type. It can take the values:

DISABLED|ENABLED

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|PURGED|INVALID|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ENTRY_NOT_FOUND

BATT gate, START_BROWSE_PROCESSTYPE function

The START_BROWSE_PROCESSTYPE function of the BATT gate is used to initiate a browse of the process types known to this region.

Input parameters

None

Output parameters

BROWSE_TOKEN
is the token used to identify this browse.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

BATT gate, GET_NEXT_PROCESSTYPE function

The GET_NEXT_PROCESSTYPE function of the BATT gate is used to return the name of the next process type in the browse, identified by the browse token.

Input parameters

BROWSE_TOKEN
is the token returned to the caller on the START_BROWSE_PROCESSTYPE call.

Output parameters

PROCESSTYPE_NAME
the 8-character process type name.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_MORE_DATA_AVAILABLE

BATT gate, END_BROWSE_PROCESSTYPE function

The END_BROWSE_PROCESSTYPE function of the BATT gate is used to end the browse identified by the browse token.

Input parameters

BROWSE_TOKEN
is the token returned to the caller on the START_BROWSE_PROCESSTYPE call.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

BATT gate, SET_PROCESSTYPE function

The SET_PROCESSTYPE function of the BATT gate is used to alter the named processtype definition.

Input parameters

PROCESSTYPE_NAME
is the 8-character process type name.

FILE_NAME is an 8-character name of the repository file to be associated with this process type.

AUDITLEVEL determines the level of auditing to be undertaken for this process type. It can take the values:

OFF|PROCESS|ACTIVITY|FULL

USERRECORDS indicates whether user audit records are to be written to the log. It can take the values:

YES|NO

STATUS indicates whether the status of the process type. It can take the values:

DISABLED|ENABLED

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ENTRY_NOT_FOUND NOT_DISABLED

BATT gate, DISCARD_PROCESSTYPE function

The DISCARD_PROCESSTYPE function of the BATT gate is used to discard the named processtype definition.

Input parameters

PROCESSTYPE_NAME
is the 8-character process type name.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ENTRY_NOT_FOUND NOT_DISABLED

BATT gate, COMMIT_PROCESSTYPE_TABLE function

The COMMIT_PROCESSTYPE_TABLE function of the BATT gate is used to commit the process type definitions to the global catalog.

Input parameters

TOKEN is the token identifying the table of process type definitions.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

BAXM gate, INIT_ACTIVITY_REQUEST function

The INIT_ACTIVITY_REQUEST function of the BAXM gate is used when the transaction requires a 3270 bridge facility, in which case the named bridge exit program is invoked.

Input parameters

REQUEST_BLOCK
a block used to hold the request data.

BRIDGE_EXIT the 8-character name of the bridge exit program.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

BAXM gate, BIND_ACTIVITY_REQUEST function

The BIND_ACTIVITY_REQUEST function of the BAXM gate is used to make the current UOW an activation of the activity specified in the activity request. This activation could be used to mark the activity complete abended because the previous activation failed, hence the abend information.

Input parameters

ABEND_CODE the 4-character abend code.

ABEND_PROG the 8-character abend program name.

ABEND_MSG the 6-character abend message number.

REQUEST_BLOCK
a block used to hold the activity request data.

Output parameters

PROGRAM is the 8-character program name.

RUN_PROGRAM is used to indicate if a program is to be invoked on the program manager INITIAL_LINK. It can take the values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND TIMEOUT READ_FAILURE

BAPR gate, ADD_PROCESS function

The ADD_PROCESS function of the BAPR gate is used to define a new process in reponse to an EXEC CICS DEFINE PROCESS call.

Input parameters

PROCESS_NAME the 36-character process name.

PROCESSTYPE the 8-character process type.

TRANID the 4-character transaction id.

PROGRAM the 8-character program name associated with the root activity.

USERID the 8-character userid.

CHECK_UNIQUE a Boolean value indicating whether a check should be made to ensure that the process name is unique within the scope of the process-type.

Output parameters

PROCESS_TOKEN
a token representing this process internally.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE_PROCESS_NAME FILE_NOT_AUTH PROCESS_ALREADY_ACQUIRED PROCESSTYPE_NOT_ENABLED PROCESSTYPE_NOT_FOUND WRITE_FAILED

BAPR gate, RUN_PROCESS function

The RUN_PROCESS function of the BAPR gate is used to execute the acquired process (invoke the root activity), either asynchronously or synchronously i.e. with a context switch.

Input parameters

MODE can take the values:

SYNC|ASYNC

INPUT_EVENT the 16-character name of the input event.

FACILITY_TOKEN
the 8-character facility token.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROCESS_NOT_FOUND PROCESSTYPE_NOT_FOUND PROCESS_SUSPENDED OTHER_PROCESS_CURRENT INVALID_EVENT INVALID_MODE AUTOINSTALL_FAILED AUTOINSTALL_INVALID_DATA AUTOINSTALL_MODEL_NOT_DEF AUTOINSTALL_URM_FAILED PROGRAM_NOT_AUTHORISED PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE REMOTE_PROGRAM SECOND_JVM_PROGRAM RUN_SYNC_ABENDED RECORD_BUSY REMOTE_TRAN TRAN_NOT_AUTH

BAPR gate, LINK_PROCESS function

The LINK_PROCESS function of the BAPR gate is used to invoke the acquired process synchronously, without a context switch.

Input parameters

INPUT_EVENT the 16-character name of the input event.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROCESS_NOT_FOUND PROCESSTYPE_NOT_FOUND PROCESS_SUSPENDED OTHER_PROCESS_CURRENT INVALID_EVENT INVALID_MODE AUTOINSTALL_FAILED AUTOINSTALL_INVALID_DATA AUTOINSTALL_MODEL_NOT_DEF AUTOINSTALL_URM_FAILED PROGRAM_NOT_AUTHORISED PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE REMOTE_PROGRAM SECOND_JVM_PROGRAM NO_EVENTS_PROCESSED PENDING_ACTIVITY_EVENTS

BAPR gate, ACQUIRE_PROCESS function

The ACQUIRE_PROCESS function of the BAPR gate is used to acquire the named process.

Input parameters

PROCESS_NAME the 36-character process name.
PROCESSTYPE the 8-character process type.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROCESS_NOT_FOUND PROCESSTYPE_NOT_FOUND FILE_NOT_AUTH OTHER_PROCESS_CURRENT RECORD_BUSY

BAPR gate, CANCEL_PROCESS function

The CANCEL_PROCESS function of the BAPR gate is used to synchronously cancel the acquired process.

Input parameters

None

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROCESS_NOT_FOUND PROCESSTYPE_NOT_FOUND FILE_NOT_AUTH RECORD_BUSY

BAPR gate, SUSPEND_PROCESS function

The SUSPEND_PROCESS function of the BAPR gate is used to suspend the acquired process.

Input parameters

None

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROCESS_NOT_FOUND RECORD_BUSY

BAPR gate, RESUME_PROCESS function

The RESUME_PROCESS function of the BAPR gate is used to resume a previously suspended process.

Input parameters

None

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROCESS_NOT_FOUND RECORD_BUSY

BAPR gate, CHECK_PROCESS function

The CHECK_PROCESS function of the BAPR gate is used to establish how the acquired process completed.

Input parameters

None

Output parameters

COMPLETION_STATUS
is the completion status of the process. It can have any of these values:

NORMAL|ABENDED|FORCEDCOMPLETE|INCOMPLETE

ABEND_CODE the 4-character abend code.

ABEND_PROGRAM
the 8-character name of the program which abended.

SUSPENDED indicates whether the process is suspended. It can take the value:
YES|NO

ACTMODE the active mode of the process. It can take the value:
INITIAL|ACTIVE|DORMANT|CANCELLING|COMPLETE

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROCESS_NOT_FOUND RECORD_BUSY

BAPR gate, REST_PROCESS function

The RESET_PROCESS function of the BAPR gate is used to reset the state of the acquired root activity to initial, so it may be run again.

Input parameters

None

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROCESS_NOT_FOUND FILE_NOT_AUTH PROCESSTYPE_NOT_FOUND INVALID_MODE RECORD_BUSY

BAAC gate, ADD_ACTIVITY function

The ADD_ACTIVITY function of the BAAC gate is used to define a new activity in response to an EXEC CICS DEFINE ACTIVITY call.

Input parameters

ACTIVITY_NAME
the 16-character activity name.

COMPLETION_EVENT
the 16-character completion event.

TRANID the 4-character transaction id.

PROGRAM the 8-character program name associated with the root activity.

USERID the 8-character userid.

ACTIVITYID the buffer containing the activity identifier.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE_ACTIVITY_NAME NO_CURRENT_ACTIVITY UNKNOWN_TRANSACTION_ID INVALID_NAME

BAAC gate, RUN_ACTIVITY function

The RUN_ACTIVITY function of the BAAC gate is used to execute the named child activity or the acquired activity either asynchronously or synchronously i.e. with a context switch.

Input parameters

ACTIVITY_NAME
the 16-character activity name.

MODE
can take the values:

SYNC|ASYNC

INPUT_EVENT
the 16-character name of the input event.

FACILITY_TOKEN
the 8-character facility token.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND INVALID_EVENT INVALID_MODE NO_CURRENT_ACTIVITY NO_COMPLETION_EVENT REMOTE_PROGRAM ACTIVITY_SUSPENDED RUN_SYNC_ABENDED READ_FAILURE RECORD_BUSY REMOTE_TRAN TRAN_NOT_AUTH

BAAC gate, LINK_ACTIVITY function

The LINK_PROCESS function of the BAAC gate is used to invoke the named child activity or acquired activity synchronously, without a context switch.

Input parameters

ACTIVITY_NAME
the 16-character name of the activity.

INPUT_EVENT
the 16-character name of the input event.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND NO_CURRENT_ACTIVITY NO_COMPLETION_EVENT INVALID_EVENT INVALID_MODE AUTOINSTALL_FAILED AUTOINSTALL_INVALID_DATA AUTOINSTALL_MODEL_NOT_DEF AUTOINSTALL_URM_FAILED PROGRAM_NOT_AUTHORISED PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE REMOTE_PROGRAM SECOND_JVM_PROGRAM NO_EVENTS_PROCESSED PENDING_ACTIVITY_EVENTS

BAAC gate, CANCEL_ACTIVITY function

The CANCEL_ACTIVITY function of the BAAC gate is used to synchronously cancel the named child activity or the acquired activity.

Input parameters

ACTIVITY_NAME
the 16-character activity name.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND NO_CURRENT_ACTIVITY INVALID_MODE INVALID_ACTIVITYID FILE_NOT_AUTH RECORD_BUSY

BAAC gate, SUSPEND_ACTIVITY function

The SUSPEND_ACTIVITY function of the BAAC gate is used to suspend the named child activity or the acquired activity.

Input parameters

ACTIVITY_NAME
the 16-character activity name.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND NO_ACQUIRED_ACTIVITY INVALID_MODE READ_FAILURE RECORD_BUSY

BAAC gate, RESUME_ACTIVITY function

The RESUME_ACTIVITY function of the BAAC gate is used to resume a previously suspended activity.

Input parameters

ACTIVITY_NAME
the 16-character activity name.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND NO_ACQUIRED_ACTIVITY INVALID_MODE READ_FAILURE RECORD_BUSY

BAAC gate, CHECK_ACTIVITY function

The CHECK_ACTIVITY function of the BAAC gate is used to establish how the named child activity or acquired activity completed.

Input parameters

ACTIVITY_NAME
the 16-character activity name.

Output parameters

COMPLETION_STATUS
is the completion status of the activity. It can have any of these values:

NORMAL|ABENDED|FORCEDCOMPLETE|INCOMPLETE

ABEND_CODE the 4-character abend code.

ABEND_PROGRAM
the 8-character name of the program which abended.

SUSPENDED indicates whether the process is suspended. It can take the value:

YES|NO

ACTMODE the active mode of the process. It can take the value:

INITIAL|ACTIVE|DORMANT|CANCELLING|COMPLETE

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND NO_CURRENT_ACTIVITY READ_FAILURE RECORD_BUSY

BAAC gate, RESET_ACTIVITY function

The RESET_ACTIVITY function of the BAAC gate is used to reset the state of the named child activity to initial, so it may be run again.

Input parameters

ACTIVITY_NAME
the 16-character activity name.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND NO_CURRENT_ACTIVITY FILE_NOT_AUTH INVALID_MODE READ_FAILURE RECORD_BUSY

BAAC gate, RETURN_END_ACTIVITY function

The RETURN_END_ACTIVITY function of the BAAC gate is used to indicate the completion of the current activity and so raise the completion event.

Input parameters

None

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY

BAAC gate, DELETE_ACTIVITY function

The DELETE_ACTIVITY function of the BAAC gate is used to delete the named child activity from the repository.

Input parameters

ACTIVITY_NAME
the 16-character activity name.

Output parameters

ACTMODE the active mode of the process. It can take the value:

INITIAL|ACTIVE|DORMANT|CANCELLING|COMPLETE

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND NO_CURRENT_ACTIVITY INVALID_MODE READ_FAILURE RECORD_BUSY

BAAC gate, ACQUIRE_ACTIVITY function

The ACQUIRE_ACTIVITY function of the BAAC gate is used to acquire the specified activity.

Input parameters

ACTIVITYID the buffer for the activity identifier.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND ACTIVITY_ALREADY_ACQUIRED READ_FAILURE RECORD_BUSY

BAAC gate, ADD_TIMER_REQUEST function

The ADD_TIMER_REQUEST function of the BAAC gate is used to add a delayed request to BAM domain in response to an EXEC CICS DEfINE TIMER call.

Input parameters

REQUEST_TOKEN
the token representing the request.

TIMER_EVENT the timer event name.

EVENT_VERSION
the version of the event.

DATETIME the time at which the timer expires.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY

BAAC gate, ADD_REATTACH_ACQUIRED function

The ADD_REATTACH_ACQUIRED function of the BAAC gate is used to reattach an activity.

Input parameters

None

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_ACQUIRED_ACTIVITY

BABR gate, STARTBR_ACTIVITY function

The STARTBR_ACTIVITY function of the BABR gate is used to initiate a browse of activities from the specified activity identifier or from the root activity of the specified process.

Input parameters

ACTIVITYID is a buffer containing the activity identifier.

PROCESS_NAME is a buffer containing the process name.

PROCESS_TYPE is the 8-character process type.

Output parameters

BROWSE_TOKEN is the token identifying the browse.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND FILE_NOT_AUTH NO_CURRENT_ACTIVITY PROCESS_NOT_FOUND PROCESSTYPE_NOT_FOUND RECORD_BUSY
INVALID	INVALID_ACTIVITYID_LEN INVALID_PROCESSNAME_LEN

BABR gate, GETNEXT_ACTIVITY function

The GETNEXT_ACTIVITY function of the BABR gate is used to return the next activity in the specified browse.

Input parameters

RETURNED_ACTIVITYID
is a buffer containing the activity identifier.

BROWSE_TOKEN is the browse token.

Output parameters

ACTIVITY_NAME
is the 16-character activity name.

LEVEL is the level into the activity tree.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END INVALID_BROWSE_TOKEN INVALID_BROWSE_TYPE RECORD_BUSY
INVALID	INVALID_BUFFER_LENGTH

BABR gate, ENDBR_ACTIVITY function

The ENDBR_ACTIVITY function of the BABR gate is used to end the specified activity browse.

Input parameters

BROWSE_TOKEN is the browse token.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN INVALID_BROWSE_TYPE

BABR gate, INQUIRE_ACTIVITY function

The INQUIRE_ACTIVITY function of the BABR gate is used to obtain information about the specified activity.

Input parameters

ACTIVITYID is a buffer containing the identifier of the activity which is to be inquired upon.

RETURNED_ACTIVITYID
is a buffer containing the returned activity identifier.

RETURNED_PROCESS_NAME
is a buffer containing the returned process name.

Output parameters

ABEND_CODE is the 4-character abend code.

ABEND_PROGRAM
is the 8-character name of the program which abended.

ACTIVITY_NAME
is the 16-character activity name.

COMPLETION_STATUS
is the completion status. It can take the values:

ABENDED|FORCED|INCOMPLETE|NORMAL

EVENT_NAME is the 16-character event name.

MODE is the mode of the activity. It can take the values:

INITIAL|ACTIVE|DORMANT|CANCELLING|COMPLETE

PROCESS_TYPE is the 8-character process type.

PROGRAM is the 8-character name of the activity program.

TRANSID is the 4-character transaction identifier.

INIT_TRANSID is the 4-character transaction identifier of the transaction under which the activity was initiated.

USERID is the 8-character userid.

SUSPENDED indicates whether the activity is currently suspended. It can take the values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND FILE_NOT_AUTH NO_CURRENT_ACTIVITY RECORD_BUSY
INVALID	INVALID_ACTIVITYID_LEN INVALID_BUFFER_LEN

BABR gate, STARTBR_CONTAINER function

The STARTBR_CONTAINER function of the BABR gate is used to initiate a browse of containers associated with a specified activity or process.

Input parameters

ACTIVITYID is a buffer containing the activity identifier.

PROCESS_NAME is a buffer containing the process name.

PROCESS_TYPE is the 8-character process type.

Output parameters

BROWSE_TOKEN is the token identifying the browse.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND FILE_NOT_AUTH NO_CURRENT_ACTIVITY PROCESS_NOT_FOUND PROCESSTYPE_NOT_FOUND RECORD_BUSY
INVALID	INVALID_ACTIVITYID_LEN INVALID_PROCESSNAME_LEN

BABR gate, GETNEXT_CONTAINER function

The GETNEXT_CONTAINER function of the BABR gate is used to return the next container in the specified browse.

Input parameters

BROWSE_TOKEN is the browse token.

Output parameters

CONTAINER_NAME
is the 16-character container name.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END INVALID_BROWSE_TOKEN INVALID_BROWSE_TYPE RECORD_BUSY

BABR gate, ENDBR_CONTAINER function

The ENDBR_CONTAINER function of the BABR gate is used to end the specified container browse.

Input parameters

BROWSE_TOKEN is the browse token.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN INVALID_BROWSE_TYPE

BABR gate, INQUIRE_CONTAINER function

The INQUIRE_CONTAINER function of the BABR gate is used to obtain information about the specified container.

Input parameters

CONTAINER_NAME
the 16-character container name.

ACTIVITYID is a buffer containing the activity identifier.

PROCESS_NAME is a buffer containing the process name.

PROCESS_TYPE is the 8-character process type.

Output parameters

DATA_LENGTH is the length of the container data.

DATA_ADDRESS is the address of the container data.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND CONTAINER_NOT_FOUND PROCESS_NOT_FOUND PROCESSTYPE_NOT_FOUND FILE_NOT_AUTH NO_CURRENT_ACTIVITY RECORD_BUSY
INVALID	INVALID_ACTIVITYID_LEN INVALID_PROCESSNAME_LEN

BABR gate, STARTBR_PROCESS function

The STARTBR_PROCESS function of the BABR gate is used to initiate a browse of the processes of a certain type.

Input parameters

PROCESS_TYPE is the 8-character process type.

Output parameters

BROWSE_TOKEN is the token identifying the browse.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	FILE_NOT_AUTH FILE_UNAVAILABLE NO_CURRENT_ACTIVITY PROCESSTYPE_NOT_FOUND RECORD_BUSY

BABR gate, GETNEXT_PROCESS function

The GETNEXT_PROCESS function of the BABR gate is used to return the next process in the specified browse.

Input parameters

BROWSE_TOKEN is the browse token.

RETURNED_ACTIVITYID
is a buffer containing the activity identifier.

RETURNED_PROCESS_NAME
is a buffer containing the process name.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END INVALID_BROWSE_TOKEN INVALID_BROWSE_TYPE RECORD_BUSY
INVALID	INVALID_BUFFER_LENGTH

BABR gate, ENDBR_PROCESS function

The ENDBR_PROCESS function of the BABR gate is used to end the specified process browse.

Input parameters

BROWSE_TOKEN is the browse token.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN INVALID_BROWSE_TYPE

BABR gate, INQUIRE_PROCESS function

The INQUIRE_PROCESS function of the BABR gate is used to obtain information about the specified process.

Input parameters

RETURNED_ACTIVITYID
is a buffer containing the activity identifier.

PROCESS_NAME is a buffer containing the process name.

PROCESS_TYPE is the 8-character process type.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROCESS_NOT_FOUND PROCESSTYPE_NOT_FOUND FILE_NOT_AUTH RECORD_BUSY
INVALID	INVALID_BUFFER_LENGTH

BABR gate, INQUIRE_ACTIVATION function

The INQUIRE_ACTIVATION function of the BABR gate is used to obtain information about the activation associated with a running transaction, if there is one.

Input parameters

TRANSACTION_TOKEN
is a token representing an instance of a transaction.

RETURNED_ACTIVITYID
is a buffer containing the activity identifier.

RETURNED_PROCESS_NAME
is a buffer containing the process name.

Output parameters

ACTIVITY_NAME
is the 16-character activity name.

PROCESS_TYPE is the 8-character process type.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND
INVALID	INVALID_BUFFER_LENGTH

BABR gate, COMMIT_BROWSE function

The COMMIT_BROWSE function of the BABR gate is used to release any CICS BTS browses associated with this UOW.

Input parameters

CHAIN_HEAD pointer to the head of the browse chain.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

BACR gate, DELETE_CONTAINER function

The DELETE_CONTAINER function of the BACR gate is used to delete a named container and its associated data.

Input parameters

CONTAINER_NAME
is the 16-character container name.

ACTIVITY_NAME
is the 16-character activity name.

CONTAINER_SCOPE
identifies the scope of this container. It can the values:

CHILD_ACTIVITY|ACTIVITY|PROCESS|
ACQUIRED_ACTIVITY|ACQUIRED_PROCESS

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND CONTAINER_NOT_FOUND NO_ACQUIRED_PROCESS NO_ACQUIRED_ACTIVITY NO_CURRENT_PROCESS NO_CURRENT_ACTIVITY RECORD_BUSY CONTAINER_READONLY

BACR gate, GET_CONTAINER_INTO function

The GET_CONTAINER_INTO function of the BACR gate is used to place the data in a named container into an area provided by the caller.

Input parameters

CONTAINER_NAME
is the 16-character container name.

ACTIVITY_NAME
is the 16-character activity name.

CONTAINER_SCOPE
identifies the scope of this container. It can have the values:

CHILD_ACTIVITY|ACTIVITY|PROCESS|
ACQUIRED_ACTIVITY|ACQUIRED_PROCESS

ITEM_BUFFER is the buffer into which the container data is placed.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND CONTAINER_NOT_FOUND LENGTH_ERROR NO_ACQUIRED_PROCESS NO_ACQUIRED_ACTIVITY NO_CURRENT_ACTIVITY NO_CURRENT_PROCESS RECORD_BUSY

BACR gate, GET_CONTAINER_LENGTH function

The GET_CONTAINER_LENGTH function of the BACR gate is used to query the length of application data in a named container.

Input parameters

CONTAINER_NAME
is the 16-character container name.

ACTIVITY_NAME
is the 16-character activity name.

CONTAINER_SCOPE
identifies the scope of this container. It can have the values:

`CHILD_ACTIVITY|ACTIVITY|PROCESS|`
`ACQUIRED_ACTIVITY|ACQUIRED_PROCESS`

Output parameters

CONTAINER_LENGTH
is the fullword length of the application data.

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND CONTAINER_NOT_FOUND NO_ACQUIRED_PROCESS NO_ACQUIRED_ACTIVITY NO_CURRENT_PROCESS NO_CURRENT_ACTIVITY

BACR gate, GET_CONTAINER_SET function

The GET_CONTAINER_SET function of the BACR gate is used to place the data in a named container into an area provided by BAM domain and return this area to the caller.

Input parameters

CONTAINER_NAME
is the 16-character container name.

ACTIVITY_NAME
is the 16-character activity name.

CONTAINER_SCOPE
identifies the scope of this container. It can the values:

CHILD_ACTIVITY|ACTIVITY|PROCESS|
ACQUIRED_ACTIVITY|ACQUIRED_PROCESS

Output parameters

ITEM_DATA a block holding the named container's data.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND CONTAINER_NOT_FOUND NO_ACQUIRED_PROCESS NO_ACQUIRED_ACTIVITY NO_CURRENT_ACTIVITY NO_CURRENT_PROCESS RECORD_BUSY

BACR gate, PUT_CONTAINER function

The PUT_CONTAINER function of the BACR gate is used to place data into a named container.

Input parameters

CONTAINER_NAME
is the 16-character container name.

ACTIVITY_NAME
is the 16-character activity name.

CONTAINER_SCOPE
identifies the scope of this container. It can the values:

CHILD_ACTIVITY|ACTIVITY|PROCESS|
ACQUIRED_ACTIVITY|ACQUIRED_PROCESS

ITEM_DATA a block holding the data to be placed in the named container.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND CONTAINER_NOT_FOUND LENGTH_ERROR NO_ACQUIRED_PROCESS NO_ACQUIRED_ACTIVITY NO_CURRENT_ACTIVITY NO_CURRENT_PROCESS INVALID_CONTAINER_NAME CONTAINER_READONLY RECORD_BUSY

BACM gate, MOVE_CONTAINER function

The MOVE_CONTAINER function of the BACM gate is used to move a container between activities. If a container of the same name as the destination container name already exists in the destination activity then it is overwritten. .

Input parameters

CONTAINER_NAME
is the 16-character source container name.

ACTIVITY_NAME
is the 16-character activity name of the activity with which the source container is associated.

CONTAINER_SCOPE
identifies the scope of the source container. It can have the values:

CHILD_ACTIVITY|ACTIVITY|PROCESS|
ACQUIRED_ACTIVITY|ACQUIRED_PROCESS

TO_ACTIVITY
is the 16-character activity name of the activity with which the destination container is associated.

AS_CONTAINER
is the 16-character destination container name.

TO_PROCESS
is a Boolean value indicating if the destination container is to be a process container rather than an activity container.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND CONTAINER_NOT_FOUND NO_ACQUIRED_PROCESS NO_ACQUIRED_ACTIVITY NO_CURRENT_PROCESS NO_CURRENT_ACTIVITY INVALID_CONTAINER_NAME CONTAINER_READONLY RECORD_BUSY

BAGD format, INQUIRE_DATA_LENGTH function

The INQUIRE_DATA_LENGTH function of the BAGD format is used by BAM domain to query the called domain as to the size of the flattened data which is to be included in the activity record.

Input parameters

DATA_TOKEN
a token representing the data.

Output parameters

DATA_LENGTH
the length of the flattened data.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|PURGED|KERNERROR|DISASTER

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_TOKEN

BAGD format, DESTROY_TOKEN function

The DESTROY_TOKEN function of the BAGD format is used by BAM domain to tell interested parties (EM domain) to destroy their data token.

Input parameters

DATA_TOKEN
a token representing the data.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|PURGED|KERNERROR|DISASTER

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_TOKEN

BAGD format, ADDRESS_DATA function

The ADDRESS_DATA function of the BAGD format is a call made to BAM domain which returns the length of the calling domain's data in the activity record.

Input parameters

ACTIVITYID
a block to hold the activity identifier.

ACQUIRED_ACTIVITY
indicates if this is an acquired activity. It can take the values:

YES|NO

Output parameters

DATA_BLOCK
a block containing the flattened data.

ACTIVITY_TOKEN
a token representing the activity.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|PURGED|KERNERROR|DISASTER

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ACTIVITY_NOT_FOUND NO_CURRENT_ACTIVITY FILE_NOT_AUTH

BAGD format, RELEASE_DATA function

The RELEASE_DATA function of the BAGD format is a call made to BAM domain which releases the calling domain's storage associated with the identified activity.

Input parameters

ACTIVITY_TOKEN
a token representing the activity.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|PURGED|KERNERROR|DISASTER

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_TOKEN

Business application manager domain’s generic gates

Table 34 summarizes the business application manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 34. Business application manager domain’s generic gates

Gate	Trace	Function	Format
DMDM	BA 0101 BA 0102	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
RMRO	BA 0140 BA 0141	PERFORM_PREPARE PERFORM_COMMIT START_BACKOUT DELIVER_BACKOUT_DATA END_BACKOUT PERFORM_SHUNT PERFRM_UNSHUNT	RMRO
RMKP	BA 0140 BA 0141	TAKE_KEYPOINT	RMKP
RMDE	BA 0140 BA 0141	START_DELIVERY DELIVER_RECOVERY END_DELIVERY	RMDE
APUE	BA 0180 BA 0181	SET_EXIT_STATUS	APUE

For descriptions of these functions and their input and output parameters, refer to the Sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—“Domain manager domain’s generic formats” on page 669
Format RMRO—“Recovery Manager domain’s call back formats” on page 1088
Format RMKP—“Recovery Manager domain’s call back formats” on page 1088
Format RMDE—“Recovery Manager domain’s call back formats” on page 1088
Format APUE—“Application domain’s generic formats” on page 593

Modules

Module	Function
DFHBADM	DFHBADM is the gate module for the following requests: PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHBATT	DFHBATT is the gate module for the following requests: ADD_REPLACE_PROCESSTYPE INQUIRE_PROCESSTYPE START_BROWSE_PROCESSTYPE GET_NEXT_PROCESSTYPE END_BROWSE_PROCESSTYPE DISCARD_PROCESSTYPE COMMIT_PROCESSTYPE_TABLE

Module	Function
DFHBAAC	DFHBAAC is the gate module for the following requests: ADD_ACTIVITY RUN_ACTIVITY CHECK_ACTIVITY RETURN_END_ACTIVITY DELETE_ACTIVITY SUSPEND_ACTIVITY RESUME_ACTIVITY CANCEL_ACTIVITY LINK_ACTIVITY ACQUIRE_ACTIVITY RESET_ACTIVITY ADD_TIMER_REQUEST ADD_REATTACH_ACQUIRED
DFHBAPR	DFHBAPR is the gate module for the following requests: ADD_PROCESS RUN_PROCESS CHECK_PROCESS SUSPEND_PROCESS RESUME_PROCESS CANCEL_PROCESS LINK_PROCESS ACQUIRE_PROCESS RESET_PROCESS
DFHBACR	DFHBACR is the gate module for the following requests: DELETE_CONTAINER GET_CONTAINER_INTO GET_CONTAINER_SET PUT_CONTAINER
DFHBAXM	DFHBAXM is the gate module for the following requests: INIT_ACTIVITY_REQUEST BIND_ACTIVITY_REQUEST
DFHBAGD	DFHBAGD is the gate module for the following requests: INQUIRE_DATA_LENGTH GET_DATA DESTROY_TOKEN ADDRESS_DATA RELEASE_DATA

Business application manager domain (BAM)

Module	Function
DFHBABR	DFHBABR is the gate module for the following requests: STARTBR_ACTIVITY GETNEXT_ACTIVITY ENDBR_ACTIVITY INQUIRE_ACTIVITY STARTBR_CONTAINER GETNEXT_CONTAINER ENDBR_CONTAINER INQUIRE_CONTAINER STARTBR_PROCESS GETNEXT_PROCESS ENDBR_PROCESS INQUIRE_PROCESS INQUIRE_ACTIVATION COMMIT_BROWSE
DFHBASP	DFHBASP is the gate module for the following requests: PERFORM_PREPARE PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT START_BACKOUT DELIVER_BACKOUT_DATA END_BACKOUT START_RECOVERY DELIVER_RECOVERY END_RECOVERY TAKE_KEYPOINT
DFHBAUE	DFHBAUE is the gate module for the following requests: SET_EXIT_STATUS
DFHBAAC0	Implements general activity class methods.
DFHBAAC1	Initialises the activity class.
DFHBAAC2	Implements the prepare method of the activity class.
DFHBAAC3	Implements the commit method of the activity class.
DFHBAAC4	Implements the delete method of the activity class.
DFHBAAC5	Implements the set_complete method of the activity class.
DFHBAAC6	Implements the invoke_exit method of the activity class.
DFHBAA10	Implements the read_activity method of the activity class.
DFHBAA11	Implements the get_activity_instance method of the activity class.
DFHBAA12	Implements the run_sync method of the activity class.
DFHBAAR1	Intialises the audit class.
DFHBAAR2	Implements the write method of the audit class.
DFHBAPR0	Implements general process class methods.
DFHBAVP1	Initialises the variable length subpool class.
DFHBAOFI	Initialises the object factory class.
DFHBABU1	Initialises the buffer class.
DFHBAPT1	Initialises the processtype class.
DFHBAPT2	Implements the rebuild_table method of the processtype class.

Module	Function
DFHBAPT3	Implements the purge_catalog method of the processtype class.
DFHBALR2	Implements the create_key method of the logical record class.
DFHBALR3	Implements the write_buffer method of the logical record class.
DFHBALR4	Implements the read_key method of the logical record class.
DFHBALR5	Implements the read_record method of the logical record class.
DFHBALR6	Implements the delete_record method of the logical record class.
DFHBALR7	Implements the get_browse_token method of the logical record class.
DFHBALR8	Implements the read_next_record method of the logical record class.
DFHBALR9	Implements the release_browse_token method of the logical record class.
DFHBARUP	The BTS repository utility program.
DFHBARUC	The BTS repository utility program.
DFHBARUD	The BTS repository utility program.
DFHBADUF	Formats the BAM domain control blocks
DFHBADU1	Formats the BAM domain control blocks
DFHBATRI	Interprets BAM domain trace entries

Exits

There are two user exit points in BAM domain, XRSINDI and XBADEACT. See the CICS Customization Guide for further details.

Trace

The point IDs for the business application manager domain are of the form BA xxxx; the corresponding trace levels are BA 1, BA 2, and Exc.

For more information about the trace points, see *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 75. CICS catalog domains (CC/GC)

The two CICS catalog domains, namely the local catalog (CC) domain and the global catalog (GC) domain, are repositories used by other domains to hold information to allow an orderly restart. They enable CICS code to read, write, and purge records on the local and global catalog data sets so that a record of the CICS state can be maintained when CICS is not running.

These domains use a common set of programs to provide a domain interface to VSAM KSDS data sets for the local catalog (DFHLCD) and for the global catalog (DFHGCD). They also conceal, from the user domain, the underlying VSAM operations.

The local catalog is initialized with the DFHCCUTL utility to contain information that is relevant to a particular CICS system, including a list of domains.

The global catalog is used to hold information that is applicable to a whole CICS system. Thus, in an XRF system consisting of one active and one alternate CICS system, there are two local catalogs and one global catalog. Conversely, in a non-XRF system, there is one local catalog and one global catalog.

The descriptions that follow relate to the common set of programs for both the local and the global catalog domains.

CICS catalog domains' specific gate

Table 35 summarizes the CICS catalog domains' specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 35. CICS catalog domains' specific gate

Gate	Trace	Function	XPI
CCCC	CC 2010	ADD	NO
	CC 2050	DELETE	NO
		GET	NO
		WRITE	NO
		GET_UPDATE	NO
		PUT_REPLACE	NO
		START_BROWSE	NO
		GET_NEXT	NO
		END_BROWSE	NO
		TYPE_PURGE	NO
		START_WRITE	NO
		WRITE_NEXT	NO
		END_WRITE	NO

The domain identifier part of the point ID, shown in the table as CC, appears in a trace as either LC (local catalog domain) or GC (global catalog domain).

In many of the functions to be described, an input parameter NAME is listed. This name is used in the construction of a VSAM key which is then used to identify a specific record in the catalog. The record may, or may not, already exist. The key is a string concatenation of the calling domain, the type, and the name. The type is a block of records for a domain. The choice of type and name for a specific domain is at the discretion of the calling domain.

CCCC gate, ADD function

The ADD function of the CCCC gate is used to add a record.

Input parameters

DATA_IN is the data to be added to the record.

TYPE identifies a block of data.

NAME is used to construct a record key, together with the domain and the type.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE INVALID_DATA_LENGTH IO_ERROR CATALOG_FULL

CCCC gate, DELETE function

The DELETE function of the CCCC gate is used to delete a record.

Input parameters

TYPE identifies a block of data.

NAME is used to construct a record key, together with the domain and the type.

[WRITE_TOKEN]
is an optional token corresponding to a START_WRITE. This avoids the need for additional connects or disconnects.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	RECORD_NOT_FOUND IO_ERROR BAD_TOKEN

CCCC gate, GET function

The GET function of the CCCC gate is used to get a record.

Input parameters

DATA_OUT If the response is OK, this contains a copy of the specified record.

TYPE identifies a block of data.

NAME is used to construct a record key, together with the domain and the type.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	RECORD_NOT_FOUND INVALID_DATA_LENGTH IO_ERROR

CCCC gate, WRITE function

The WRITE function of the CCCC gate is used to write a record.

Input parameters

DATA_OUT is the data to be written to the specified record.
TYPE identifies a block of data.
NAME is used to construct a record key, together with the domain and the type.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_DATA_LENGTH IO_ERROR CATALOG_FULL

CCCC gate, GET_UPDATE function

The GET_UPDATE function of the CCCC gate is used to get a record and to establish a thread. This thread, identified by a token, is used in a corresponding PUT_REPLACE.

Input parameters

DATA_OUT If response is OK, this contains a copy of the record.
TYPE identifies a block of data.
NAME is used to construct a record key, together with the domain and the type.

Output parameters

UPDATE_TOKEN Token to be used by the corresponding PUT_REPLACE.
RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	RECORD_NOT_FOUND INVALID_DATA_LENGTH IO_ERROR

CCCC gate, PUT_REPLACE function

The PUT_REPLACE function of the CCCC gate is used to replace a record.

Input parameters

DATA_IN is the data to be copied to the record.
UPDATE_TOKEN is the token obtained from a previous GET_UPDATE, used to identify an existing record in the catalog.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BAD_TOKEN INVALID_DATA_LENGTH IO_ERROR CATALOG_FULL

CCCC gate, START_BROWSE function

The START_BROWSE function of the CCCC gate is used to start a browse session.

Input parameters

TYPE identifies a block of data. The browse positions itself before the first record for that type.

Output parameters

BROWSE_TOKEN is the token identifying this browse session.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. It has this value:

IO_ERROR

CCCC gate, GET_NEXT function

The GET_NEXT function of the CCCC gate is used to get the next record.

Input parameters

BROWSE_TOKEN is the token identifying this browse session.

DATA_OUT is a copy of the next record within the browsed type.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_DATA_LENGTH BAD_TOKEN BROWSE_END IO_ERROR

CCCC gate, END_BROWSE function

The END_BROWSE function of the CCCC gate is used to end a browse session.

Input parameters

BROWSE_TOKEN is the token identifying this browse session.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BAD_TOKEN IO_ERROR

CCCC gate, TYPE_PURGE function

The TYPE_PURGE function of the CCCC gate is used to purge records. This deletes all records within the specified TYPE block for that domain.

Input parameters

TYPE identifies a block of data.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TYPE_NOT_FOUND IO_ERROR

CCCC gate, START_WRITE function

The START_WRITE function of the CCCC gate is used to start a write session.

Input parameters

None.

Output parameters

WRITE_TOKEN is the token identifying a unique file string (thread).

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. It has this value:

IO_ERROR

CCCC gate, WRITE_NEXT function

The WRITE_NEXT function of the CCCC gate is used to write the next record.

Input parameters

WRITE_TOKEN is the token corresponding to the token from START_WRITE.

DATA_IN is the data to be copied to the record.

TYPE identifies a block of data.

NAME is used to construct a record key, together with the domain and the type.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_DATA_LENGTH IO_ERROR CATALOG_FULL BAD_TOKEN

CCCC gate, END_WRITE function

The END_WRITE function of the CCCC gate is used to end a write session.

Input parameters

WRITE_TOKEN Token corresponding to a START_WRITE.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IO_ERROR BAD_TOKEN

CICS catalog domains’ generic gate

Table 36 summarizes the CICS catalog domains’ generic gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic formats for calls to the gates.

Table 36. CICS catalog domains’ generic gate

Gate	Trace	Function	Format
DMDM	CC 1010 CC 1040	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

The domain identifier part of the point ID, shown in the table as CC, appears in a trace as either LC (local catalog domain) or GC (global catalog domain).

Descriptions of these functions and their input and output parameters are given in the section dealing with the corresponding generic formats. This is in format DMDM—see “Domain manager domain’s generic formats” on page 669.

In preinitialization processing, the local catalog domain opens the CICS local catalog, DFHLCD. (There is no preinitialization processing for the global catalog domain.)

In initialization processing, the global catalog domain opens the CICS global catalog, DFHGCD.

In quiesce processing, the local and global catalog domains close their respective catalog data sets.

In termination processing, the CICS catalog domains perform no termination processing. They do not close either the local catalog or the global catalog; the operating system closes these data sets.

Modules

Module	Function
DFHCCCC	Handles the following functions: ADD DELETE GET WRITE GET_UPDATE PUT_REPLACE START_BROWSE GET_NEXT END_BROWSE TYPE_PURGE START_WRITE WRITE_NEXT END_WRITE
DFHCCDM	Handles the initialization and termination of the CICS catalog domains.
DFHCCDUF	Catalog dump formatting routine.

Module	Function
DFHCCTRI	Trace interpreter routine for the catalog domains.
DFHCCUTL	Offline utility to initialize the local catalog.

Exits

No global user exit points are provided in these domains.

Trace

The point IDs for the local catalog domain are of the form LC xxxx; the corresponding trace levels are LC 1 and Exc.

The point IDs for the global catalog domain are of the form GC xxxx; the corresponding trace levels are GC 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 76. Directory manager domain (DD)

The directory manager domain (also sometimes known simply as “directory manager”) manages directories of named tokens.

Directory manager domain’s specific gates

Table 37 summarizes the directory manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, and the functions provided by the gates.

Table 37. Directory manager domain’s specific gates

Gate	Trace	Function
DDDI	DD 0201 DD 0202	CREATE_DIRECTORY ADD_ENTRY DELETE_ENTRY REPLACE_DATA
DDLO	DD 0301 DD 0302	LOCATE
DDBR	DD 0401 DD 0402	START_BROWSE GET_NEXT_ENTRY END_BROWSE

DDDI gate, CREATE_DIRECTORY function

The CREATE_DIRECTORY function of the DDDI gate is used to create a new directory with entry names of a given length.

Input parameters

DIRECTORY_NAME
is the four_character name of the directory to be created.

NAME_LENGTH
is the length of entry names in the directory. This value must be a multiple of four, and less than 256.

Output parameters

DIRECTORY_TOKEN
is the directory token

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	DUPLICATE_DIRECTORY INVALID_NAME_LEN

DDDI gate, ADD_ENTRY function

The ADD_ENTRY function of the DDDI gate is used to add an entry to a directory.

Input parameters

DIRECTORY_TOKEN
is the token for the directory.

ENTRY_NAME
is the address of the entry name. The length is fixed for the directory.

DATA_TOKEN
is the data to be associated with the entry name in the directory.

SUSPEND
indicates whether Storage Manager GETMAIN requests should be conditional or unconditional. Takes one of the values:

YES|NO

Output parameters

DUPLICATE_DATA_TOKEN
is the data currently associated with the entry name if it already exists in the directory.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE INSUFFICIENT_STORAGE
INVALID	INVALID_DIRECTORY

DDDI gate, DELETE_ENTRY function

The DELETE_ENTRY function of the DDDI gate is used to delete an entry from a directory.

Input parameters

DIRECTORY_TOKEN
is the token for the directory.

ENTRY_NAME
is the address of the entry name. The length is fixed for the directory.

Output parameters

DATA_TOKEN
is the data associated with the entry name when it was deleted.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
INVALID	INVALID_DIRECTORY

DDDI gate, REPLACE_DATA function

The REPLACE_DATA function of the DDDI gate is used to replace the data associated with an existing entry name in a directory.

Input parameters

DIRECTORY_TOKEN
is the token for the directory.

ENTRY_NAME
is the address of the entry name. The length is fixed for the directory.

NEW_DATA_TOKEN
is the new data to be associated with the entry name.

PRIOR_DATA_TOKEN
is an optional parameter that indicates the data expected to be associated with the entry name just prior to it being replaced.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND DATA_CHANGED
INVALID	INVALID_DIRECTORY

DDLO gate, LOCATE function

The LOCATE function of the DDLO gate is used to locate the data associated with an existing entry name in a directory.

Input parameters

DIRECTORY_TOKEN
is the token for the directory.

ENTRY_NAME
is the address of the entry name. The length is fixed for the directory.

Output parameters

DATA_TOKEN
is the data associated with the entry name.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
INVALID	INVALID_DIRECTORY

DDBR gate, START_BROWSE function

The START_BROWSE function of the DDBR gate is used to start an alphabetical browse through all of the entries in a directory.

Input parameters

DIRECTORY_TOKEN
is the token for the directory.

AT_NAME
is the address of an entry name at which the browse is to start. The first name found will be the first which is greater than or equal to this in alphabetical order.

TASK_RELATED
is an optional parameter which indicates whether the browse will end at task end. It can be one of these values:

YES|NO

if not specified this parameter defaults to YES.

Output parameters

BROWSE_TOKEN
is the token for this browse.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_DIRECTORY

DDBR gate, GET_NEXT_ENTRY function

The GET_NEXT_ENTRY function of the DDBR gate is used to get the next entry name in alphabetical order in a directory.

Input parameters

DIRECTORY_TOKEN
is the token for the directory.

BROWSE_TOKEN
is the token for the browse.

ENTRY_NAME
is a buffer in which the entry name will be returned.

Output parameters

DATA_TOKEN
is the token associated with the entry name.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END
INVALID	INVALID_DIRECTORY INVALID_BROWSE INVALID_NAME

DDBR gate, END_BROWSE function

The END_BROWSE function of the DDBR gate is used to end a browse on a directory.

Input parameters

DIRECTORY_TOKEN
is the token for the directory.

BROWSE_TOKEN
is the token for the browse.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_DIRECTORY INVALID_BROWSE

Directory manager domain's generic gates

Table 38 summarizes the directory manager domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 38. Directory manager domain's generic gates

Gate	Trace	Function	Format
DDDM	DD 0101 DD 0102	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669

In preinitialization the directory manager adds its general subpool and global lock.

In initialization, quiesce, and termination processing, the directory manager domain performs only internal routines.

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the directory manager domain are of the form DD xxxx; the corresponding trace levels are DD 1, DD 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 77. Document Handler domain (DH)

The document handler domain manages CICS Documents.

Document Handler domain's specific gates

Table 39 summarizes the document handler domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 39. Document Handler domain's specific gates

Gate	Trace	Function	XPI
DHDH	DH 0120 DH 0121	CREATE_DOCUMENT INSERT_DATA INSERT_BOOKMARK REPLACE_DATA DELETE_DOCUMENT DELETE_DATA DELETE_BOOKMARK RETRIEVE_WITH_CTLINFO RETRIEVE_WITHOUT_CTLINFO INQUIRE_DOCUMENT	NO NO NO NO NO NO NO NO NO NO
DHSL	DH 0200 DH 0201	SET_SYMBOL_VALUE_BY_API, SET_SYMBOL_VALUE_BY_SSI, ADD_SYMBOL_LIST EXPORT_SYMBOL_LIST IMPORT_SYMBOL_LIST	NO NO NO NO NO
DHTM	DH 0401 DH 0402	INITIALIZE_DOCTEMPLATES ADD_REPLACE_DOCTEMPLATE DELETE_DOCTEMPLATE INQUIRE_DOCTEMPLATE INQUIRE_TEMPLATE_STATUS START_BROWSE GET_NEXT END_BROWSE READ_TEMPLATE	NO NO NO NO NO NO NO NO NO
DHRP	DH 0C01 DH 0C02	RECOVER_DEFINITIONS	NO

DHDH gate, CREATE_DOCUMENT function

The CREATE_DOCUMENT function of the DHDH gate is used to create a new CICS document.

Input parameters

[TEXT] is a buffer containing a block of text to be added to the document.

[BINARY] is a buffer containing a block of binary data to be added to the document.

[TEMPLATE_BUFFER]
is a buffer containing a template to be added to the document.

[TEMPLATE_NAME]
is the name of an RDO defined DOCTEMPLATE which is to be added to the document.

[SOURCE_DOCUMENT]
is the document token of an existing document created by the same CICS task which is to be added to the document.

[RETRIEVED_DOCUMENT]
is a buffer containing a document in a retrieved format which is to be added to the document.

[HOST_CODEPAGE]
is the character encoding for the block of data being added to the document. This parameter is taken into account for the TEXT and TEMPLATE_BUFFER options and ignored for all other options.

[SYMBOL_LIST]
is a buffer containing a list of symbols to be added to the symbol table of the document.

[TEMPLATE_IN_ERROR]
is a buffer which is used by the Document Handler domain to return the name of a DOCTEMPLATE in which an error has been detected. This parameter is only meaningful when specified with the TEMPLATE_NAME option or the TEMPLATE_BUFFER option where the template in the TEMPLATE_BUFFER option contains an embedded template.

Output parameters

DOCUMENT_TOKEN
is the token identifying the newly created document.

ERROR_OFFSET
is the offset into a template where a syntax error has been detected.

RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CODEPAGE_NOT_SPECIFIED INVALID_HOST_CODEPAGE INVALID_TEMPLATE_SYNTAX TEMPLATE_NOT_FOUND SOURCE_DOC_NOT_FOUND INVALID_RETRIEVE_FORMAT SYMBOL_NAME_INVALID SYMBOL_VALUE_INVALID EMBED_DEPTH_EXCEEDED INVALID_TEMPLATE_LENGTH

DHDH gate, INSERT_DATA function

The INSERT_DATA function of the DHDH gate is used to insert a block of data into an existing document.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document into which the data will be inserted.

[TEXT]
is a buffer containing a block of text to be added to the document.

[BINARY]
is a buffer containing a block of binary data to be added to the document.

[TEMPLATE_BUFFER]
is a buffer containing a template to be added to the document.

[TEMPLATE_NAME]
is the name of an RDO defined DOCTEMPLATE which is to be added to the document.

[SYMBOL]
is the name of a symbol defined in the symbol table. The value associated with the symbol will be added to the document.

[SOURCE_DOCUMENT]
is the document token of an existing document created by the same CICS task which is to be added to the document.

[RETRIEVED_DOCUMENT]
is a buffer containing a document in a retrieved format which is to be added to the document.

[HOST_CODEPAGE]
is the character encoding for the block of data being added to the document. This parameter is taken into account for the TEXT, SYMBOL and TEMPLATE_BUFFER options and ignored for all other options.

[INSERT_POINT]
identifies the beginning or end as the position at which data should be inserted into a document. It can have either of these values:

START|END

[INSERT_AT]
is the name of a bookmark which identifies the position at which the data should be inserted.

[TEMPLATE_IN_ERROR]
is a buffer which is used by the Document Handler domain to return the name of a DOCTEMPLATE in which an error has been detected. This parameter is only meaningful when specified with the TEMPLATE_NAME option or the TEMPLATE_BUFFER option where the template in the TEMPLATE_BUFFER option contains an embedded template.

Output parameters

ERROR_OFFSET
is the offset into a template where a syntax error has been detected.

RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CODEPAGE_NOT_SPECIFIED INVALID_HOST_CODEPAGE EMBED_DEPTH_EXCEEDED INSERTPOINT_NOT_FOUND INVALID_TEMPLATE_SYNTAX TEMPLATE_NOT_FOUND SOURCE_DOC_NOT_FOUND INVALID_RETRIEVE_FORMAT SYMBOL_NOT_FOUND INVALID_TEMPLATE_LENGTH
INVALID	DOCUMENT_NOT_FOUND

DHDH gate, INSERT_BOOKMARK function

The INSERT_BOOKMARK function of the DHDH gate is used to insert a bookmark into an existing document.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document into which the bookmark will be inserted.

BOOKMARK_NAME
is the 16 byte name of a bookmark to be added to the document.

[INSERT_POINT]
identifies the beginning or end as the position at which the bookmark should be inserted into a document. It can have either of these values:

START|END

[INSERT_AT]
is the name of a bookmark which identifies the position at which the bookmark should be inserted.

Output parameters

RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INSERTPOINT_NOT_FOUND INVALID_BOOKMARK_NAME DUPLICATE_BOOKMARK
INVALID	DOCUMENT_NOT_FOUND

DHDH gate, REPLACE_DATA function

The REPLACE_DATA function of the DHDH gate is used to replace the data between 2 bookmarks in an existing document.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document into which the data will be inserted.

[TEXT] is a buffer containing a block of text to be added to the document.

[BINARY] is a buffer containing a block of binary data to be added to the document.

[TEMPLATE_BUFFER]
is a buffer containing a template to be added to the document.

[TEMPLATE_NAME]
is the name of an RDO defined DOCTEMPLATE which is to be added to the document.

[SYMBOL] is the name of a symbol defined in the symbol table. The value associated with the symbol will be added to the document.

[SOURCE_DOCUMENT]
is the document token of an existing document created by the same CICS task which is to be added to the document.

[RETRIEVED_DOCUMENT]
is a buffer containing a document in a retrieved format which is to be added to the document.

[HOST_CODEPAGE]
is the character encoding for the block of data being added to the document. This parameter is taken into account for the TEXT, SYMBOL and TEMPLATE_BUFFER options and ignored for all other options.

[FROM_POSITION]
identifies the beginning or end of the document as the start of the data which is to be replaced in the document. It can have either of these values:

START|END

[FROM_BOOKMARK]
is the name of a bookmark which identifies the start of the data which is to be replaced.

[TO_POSITION]
identifies the beginning or end of the document as the end of the data which is to be replaced in the document. It can have either of these values:

START|END

[TO_BOOKMARK]
is the name of a bookmark which identifies the end of the data which is to be replaced.

[TEMPLATE_IN_ERROR]
is a buffer which is used by the Document Handler domain to return the name of a DOCTEMPLATE in which an error has been detected. This parameter is only meaningful when specified with the TEMPLATE_NAME option or the TEMPLATE_BUFFER option where the template in the TEMPLATE_BUFFER option contains an embedded template.

Output parameters

ERROR_OFFSET is the offset into a template where a syntax error has been detected.

RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CODEPAGE_NOT_SPECIFIED INVALID_HOST_CODEPAGE EMBED_DEPTH_EXCEEDED INVALID_TEMPLATE_SYNTAX TEMPLATE_NOT_FOUND SOURCE_DOC_NOT_FOUND INVALID_RETRIEVE_FORMAT SYMBOL_NOT_FOUND FROM_BOOKMARK_NOT_FOUND TO_BOOKMARK_NOT_FOUND INVALID_TEMPLATE_LENGTH
INVALID	DOCUMENT_NOT_FOUND

DHDH gate, DELETE_DOCUMENT function

The DELETE_DOCUMENT function of the DHDH gate is used to delete a document.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document to be deleted.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	DOCUMENT_NOT_FOUND

DHDH gate, DELETE_DATA function

The DELETE_DATA function of the DHDH gate is used to delete the data between 2 bookmarks in an existing document.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document from which the data will be deleted.

[FROM_POSITION]
identifies the beginning or end of the document as the start of the data which is to be deleted from the document. It can have either of these values:

START|END

[FROM_BOOKMARK]
is the name of a bookmark which identifies the start of the data which is to be deleted.

[TO_POSITION]
identifies the beginning or end of the document as the end of the data which is to be deleted from the document. It can have either of these values:

START|END

[TO_BOOKMARK]
is the name of a bookmark which identifies the end of the data which is to be deleted.

Output parameters

RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	FROM_BOOKMARK_NOT_FOUND TO_BOOKMARK_NOT_FOUND INVALID_BOOKMARK_SEQUENCE
INVALID	DOCUMENT_NOT_FOUND

DHDH gate, DELETE_BOOKMARK function

The DELETE_BOOKMARK function of the DHDH gate is used to delete a bookmark in an existing document.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document from which the bookmark will be deleted.

BOOKMARK_NAME
is the name of the bookmark to be deleted from the document.

Output parameters

RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BOOKMARK_NOT_FOUND
INVALID	DOCUMENT_NOT_FOUND

DHDH gate, RETRIEVE_WITH_CTLINFO function

The RETRIEVE_WITH_CTLINFO function of the DHDH gate is used to retrieve a copy of an existing document. The retrieved copy will contain embedded control information.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document to be retrieved.

DOCUMENT_BUFFER
is a buffer into which the Document Handler domain will place the copy of the document.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	DOCUMENT_NOT_FOUND

DHDH gate, RETRIEVE_WITHOUT_CTLINFO function

The RETRIEVE_WITHOUT_CTLINFO function of the DHDH gate is used to retrieve a copy of an existing document. The retrieved copy will only contain the data in the document.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document to be retrieved.

DOCUMENT_BUFFER
is a buffer into which the Document Handler domain will place the copy of the document.

[CLIENT_CODEPAGE]
is the character encoding that the retrieved document should be converted to when it is placed in the buffer.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_HOST_CODEPAGE INVALID_CLIENT_CODEPAGE
INVALID	DOCUMENT_NOT_FOUND

DHDH gate, INQUIRE_DOCUMENT function

The INQUIRE_DOCUMENT function of the DHDH gate is used to obtain information about the document.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document to be queried.

Output parameters

[DOCUMENT_SIZE]
is the size of the data in a document.

[RETRIEVE_SIZE]
is the maximum size in bytes that a retrieved copy of the document can be.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	DOCUMENT_NOT_FOUND

DHSL gate, SET_SYMBOL_VALUE_BY_API function

The SET_SYMBOL_VALUE_BY_API function of the DHSL gate is used to set the value of a symbol in the symbol table. If the symbol does not exist in the table, it will be added. If the symbol does exist in the table, it will always be replaced.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document that owns the symbol table.

SYMBOL_NAME
is the name of the symbol in the symbol table.

VALUE
is the value to be associated with the symbol.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	GETMAIN_ERROR SYMBOL_NAME_INVALID
INVALID	DOCUMENT_NOT_FOUND

DHSL gate, SET_SYMBOL_VALUE_BY_SSI function

The SET_SYMBOL_VALUE_BY_SSI function of the DHSL gate is used to set the value of a symbol in the symbol table. If the symbol does not exist in the table, it will be added. If the symbol does exist in the table, it will only be replaced if it was previously set using the SET_SYMBOL_VALUE_BY_SSI function.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document that owns the symbol table.

SYMBOL_NAME is the name of the symbol in the symbol table.

VALUE is the value to be associated with the symbol.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	GETMAIN_ERROR SYMBOL_NAME_INVALID
INVALID	DOCUMENT_NOT_FOUND

DHSL gate, ADD_SYMBOL_LIST function

The ADD_SYMBOL_LIST function of the DHSL gate is used to add a list of symbols to the symbol table at one time.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document that owns the symbol table.

SYMBOL_LIST is a buffer containing a list of symbols to be added to the symbol table of the document.

Output parameters

ERROR_OFFSET is the offset into the symbol list where a syntax error has been detected.

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	GETMAIN_ERROR SYMBOL_NAME_INVALID SYMBOL_VALUE_INVALID
INVALID	DOCUMENT_NOT_FOUND

DHSL gate, EXPORT_SYMBOL_LIST function

The EXPORT_SYMBOL_LIST function of the DHSL gate is used to export all the symbols in the symbol table in a form that can be re-imported with IMPORT_SYMBOL_LIST.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document that owns the symbol table.

SYMBOL_LIST_BUFFER
is a buffer that is to contain the exported symbol list.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	DOCUMENT_NOT_FOUND

DHSL gate, IMPORT_SYMBOL_LIST function

The IMPORT_SYMBOL_LIST function of the DHSL gate is used to import all the symbols in the symbol table that were exported with EXPORT_SYMBOL_LIST.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document that owns the symbol table.

SYMBOL_LIST_BUFFER
is a buffer that contains the symbol list to be added to the symbol table. This list should have been created using and the EXPORT_SYMBOL_LIST function.

Output parameters

ERROR_OFFSET
is the offset into the list where a syntax error has been detected.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SYMBOL_NAME_INVALID SYMBOL_VALUE_INVALID
INVALID	DOCUMENT_NOT_FOUND

DHTM gate, INITIALIZE_DOCTEMPLATES function

The INITIALIZE_DOCTEMPLATES function of the DHSL gate is used to initialize the state required by the template manager.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DIRECTORY_ERROR

DHTM gate, ADD_REPLACE_DOCTEMPLATE function

The ADD_REPLACE_DOCTEMPLATE function of the DHTM gate is used to install a document template into the currently executing CICS system.

Input parameters

DOCTEMPLATE
is the name of the DOCTEMPLATE resource that is to be added.

TEMPLATE_NAME
is the name by which the DOCTEMPLATE is known outside of RDO.

RESOURCE_TYPE
specifies the type of resource containing the DOCTEMPLATE. It can have one of the following values:

PDS_MEMBER|FILE|PROGRAM|TSQUEUE|TDQUEUE|EXITPGM

RESOURCE_NAME
is the name of the resource containing the DOCTEMPLATE.

[DDNAME]
is the DDNAME of the PDS containing the DOCTEMPLATE resource if the resource resides on a PDS.

Output parameters

[DATASET]
is the dataset name of the PDS containing the DOCTEMPLATE resource if the resource resides on a PDS.

[DOCTEMPLATE_IN_USE]
is the name of the DOCTEMPLATE definition that uses the same TEMPLATE_NAME as the resource being defined.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID, DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_RESOURCE_TYPE
DISASTER	DIRECTORY_ERROR
EXCEPTION	GETMAIN_FAILED NAME_IN_USE NOT_FOUND DDNAME_NOT_FOUND MEMBER_NOT_FOUND

DHTM gate, READ_TEMPLATE function

The READ_TEMPLATE function of the DHTM gate is used to read a named template into a buffer provided by the caller.

Input parameters

TEMPLATE_NAME
is the name of a previously installed document template.

TEMPLATE_BUFFER
is the buffer into which the template is to be read.

Output parameters

[DOCTEMPLATE]
is the name of the DOCTEMPLATE resource as it is known to RDO.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DIRECTORY_ERROR INVALID_RESOURCE_TYPE
EXCEPTION	NOT_FOUND NOT_USABLE TRUNCATED

DHTM gate, INQUIRE_DOCTEMPLATE function

The INQUIRE_DOCTEMPLATE function of the DHTM gate returns information about a previously installed document template.

Input parameters

DOCTEMPLATE is the name of the DOCTEMPLATE as known to RDO.

Output parameters

TEMPLATE_NAME
is the full name of the template as known outside RDO.

RESOURCE_TYPE
is the CICS or non-CICS resource type associated with the template. It can have one of the following values:

PDS_MEMBER|FILE|PROGRAM|TSQUEUE|TDQUEUE|EXITPGM

RESOURCE_NAME
is the name of the CICS or non-CICS resource.

DATASET is the dataset name of the template PDS if the RESOURCE_TYPE indicates a PDS.

DDNAME is the DDNAME of the template PDS if the RESOURCE_TYPE indicates a PDS.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DIRECTORY_ERROR
EXCEPTION	NOT_FOUND

DHTM gate, INQUIRE_TEMPLATE_STATUS function

The INQUIRE_TEMPLATE_STATUS function of the DHTM gate is used to inquire the install status of one or more templates.

Input parameters

TEMPLATE_NAME_LIST
is a list of template names whose install status is sought.

TEMPLATE_STATUS_LIST
is a list of install status indicators for the templates named in the TEMPLATE_NAME_LIST

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

DHTM gate, DELETE_DOCTEMPLATE function

The DELETE_DOCTEMPLATE function of the DHTM gate deletes a previously installed DOCTEMPLATE.

Input parameters

DOCTEMPLATE is the name of the DOCTEMPLATE as known to RDO.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DIRECTORY_ERROR
EXCEPTION	NOT_FOUND

DHTM gate, START_BROWSE function

The START_BROWSE function of the DHTM gate is used to initiate a browse of installed DOCTEMPLATE definitions.

Output parameters

BROWSE_TOKEN is a token identifying this DOCTEMPLATE browse.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

DHTM gate, GET_NEXT function

The GET_NEXT function of the DHTM gate returns information about the next installed DOCTEMPLATE in the browse.

Input parameters

BROWSE_TOKEN is the token identifying this browse of the DOCTEMPLATE definitions.

Output parameters

DOCTEMPLATE is the name of the DOCTEMPLATE as known to RDO.

TEMPLATE_NAME
is the full name of the template as known outside RDO.

RESOURCE_TYPE
is the CICS or non-CICS resource type associated with the template. It can have one of the following values:

PDS_MEMBER|FILE|PROGRAM|TSQUEUE|TDQUEUE|EXITPGM

RESOURCE_NAME
is the name of the CICS or non-CICS resource.

DATASET is the dataset name of the template PDS if the RESOURCE_TYPE indicates a PDS.

DDNAME is the DDNAME of the template PDS if the RESOURCE_TYPE indicates a PDS.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_BROWSE_TOKEN
EXCEPTION	BROWSE_END

DHTM gate, END_BROWSE function

The END_BROWSE function of the DHTM gate is used to terminate a browse of installed DOCTEMPLATE definitions.

Input parameters

BROWSE_TOKEN is the token identifying this browse of the DOCTEMPLATE definitions.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_BROWSE_TOKEN

DHRP gate, RECOVER_DEFINITIONS function

The RECOVER_DEFINITIONS function of the DHRP gate is used to purge/recover DOCTEMPLATE definitions from the global catalog depending upon the CICS start type.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	INVALID_BROWSE_TOKEN CATALOG_BROWSE_FAILURE CATALOG_PURGE_FAILURE LOGIC_ERROR WAIT_PHASE_FAILURE ABEND

Document Handler domain's generic gates

Table 40 summarizes the document handler domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 40. Document Handler domain's generic gates

Gate	Trace	Function	Format
DMDM	DH 0101 DH 0102 DH 0103 DH 0104 DH 0105 DH 0106 DH 0107 DH 0108	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
APUE	DH 0D01 DH 0D02 DH 0D03 DH 0D04 DH 0D05 DH 0D06 DH 0D07 DH 0D08	SET_EXIT_STATUS	APUE
RMRO	DH 0301 DH 0302 DH 0303 DH 0304 DH 0305 DH 0308	PERFORM_PREPARE PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT START_BACKOUT END_BACKOUT	RMRO

Table 40. Document Handler domain's generic gates (continued)

Gate	Trace	Function	Format
RMDE	DH 0301 DH 0302 DH 0303 DH 0304 DH 0306 DH 0308	START_DELIVERY DELIVER_RECOVERY END_DELIVERY	RMDE
RMKP	DH 0301 DH 0302 DH 0303 DH 0304 DH 0307 DH 0308	TAKE_KEYPOINT	RMKP

For descriptions of these functions and their input and output parameters, refer to the Sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669

Format APUE—"Application domain's generic formats" on page 593

Format RMRO—"Recovery Manager domain's call back formats" on page 1088

Format RMDE—"Recovery Manager domain's call back formats" on page 1088

Format RMKP—"Recovery Manager domain's call back formats" on page 1088

Modules

Module	Function
DFHDHDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHDHDH	Handles the following requests: CREATE_DOCUMENT INSERT_DATA INSERT_BOOKMARK REPLACE_DATA DELETE_DOCUMENT DELETE_DATA DELETE_BOOKMARK RETRIEVE_WITH_CTLINFO RETRIEVE_WITHOUT_CTLINFO INQUIRE_DOCUMENT
DFHDHSL	Handles the following requests: SET_SYMBOL_VALUE_BY_API, SET_SYMBOL_VALUE_BY_SSI, ADD_SYMBOL_LIST EXPORT_SYMBOL_LIST IMPORT_SYMBOL_LIST

Module	Function
DFHDHTM	Handles the following requests: INITIALIZE_DOCTEMPLATES ADD_REPLACE_DOCTEMPLATE DELETE_DOCTEMPLATE INQUIRE_DOCTEMPLATE INQUIRE_TEMPLATE_STATUS START_BROWSE GET_NEXT END_BROWSE READ_TEMPLATE
DFHDHRM	Handles the following requests: PERFORM_PREPARE PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT START_BACKOUT END_BACKOUT START_DELIVERY DELIVER_RECOVERY END_DELIVERY TAKE_KEYPOINT
DFHDHUE	Handles the following requests: SET_EXIT_STATUS
DFHDHPB	Processes data supplied on the BINARY parameter of CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPD	Processes data supplied on the SOURCE_DOCUMENT parameter of CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPM	Processes data supplied on the TEMPLATE_NAME parameter of CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPS	Processes data supplied on the SYMBOL parameter of INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPT	Processes data supplied on the TEXT parameter of CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPU	Processes data supplied on the TEMPLATE_BUFFER parameter of CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPX	Processes data supplied on the RETRIEVED_DOCUMENT parameter of CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPR	Reads templates held as member's of partitioned datasets.
DFHDHEI	Reads templates held on CICS resources.
DFHDHPR	Reads PDS members containing templates.
DFHDHDUF	DH domain offline dump formatting routine
DFHDHTRI	Interprets DH domain trace entries

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the document handler domain are of the form DH xxxx; the corresponding trace levels are DH 1, DH ,.2, and Exc.

Document Handler domain (DH)

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 78. Domain manager domain (DM)

The domain manager domain (also sometimes known simply as “domain manager”) is responsible for maintaining, through the use of catalog services, permanent information about individual domains.

Each domain has certain permanent characteristics. These are stored on the local catalog and include the name, token, and ID; these characteristics are unique for each domain. Each domain also has volatile characteristics (including the phase number and the status), which are not stored on the catalog.

The domain manager attaches initialization and termination tasks for other domains. It maintains phase information of the other domains to allow controlled introduction and withdrawal of domain services during initialization and termination. For each domain, a phase number denotes the set of services that are available from the domain. An increased phase number would correspond to an increased set of available functions.

During initialization, the system phase is the minimum of the phase numbers of the active domains. During shutdown, the system phase is the maximum of the phase numbers of the active domains.

The domain manager also maintains and manages a queue of waiting domains (called “waiters”); these waiting domains are waiting for a specific domain to reach a certain phase or for the system phase to reach a certain level.

Domain manager domain’s specific gates

Table 41 summarizes the domain manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 41. Domain manager domain’s specific gates

Gate	Trace	Function	XPI
DMDM	DM 0001	ADD_DOMAIN	NO
	DM 0002	QUIESCE_SYSTEM	NO
		SET_PHASE	NO
		WAIT_PHASE	NO
DMIQ	DM 0003	START_BROWSE	NO
	DM 0004	GET_NEXT	NO
		END_BROWSE	NO
		INQ_DOMAIN_BY_NAME	NO
		INQ_DOMAIN_BY_TOKEN	NO
		INQ_DOMAIN_BY_ID	NO
DMEN	DM 0210	LISTEN	NO
	DM 0211	DELETE	NO
		NOTIFY_SMSVSAM_OPERATIONAL	NO

DMEN gate, LISTEN function

The LISTEN function of the DMEN gate is issued to register an interest in an event notification facility (ENF) event. The MVS event notification facility is a generalized communication facility which allows subsystems to broadcast notification of events.

If a domain wishes to be notified of particular ENF events, it must register the events that it wishes to be notified of with Domain Manager using the LISTEN interface.

When an ENF event occurs domain manager will invoke the named listen gate of all domains that registered for that event.

Input parameters

EVENT is the event that the caller is registering an interest in, and can have any of these values:

SMSVSAM_OPERATIONAL

LISTEN_GATE is the gate number of the gate at which the caller wishes to be notified when the event occurs.

Output parameters

RESPONSE is DFHDMEN's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when response is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_EVENT DUPLICATE_LISTEN

DMEN gate, DELETE function

The DELETE function of the DMEN gate is used to deregister an interest in an ENF event.

If a domain is registered with domain manager for notification of an ENF event and that domain no longer wishes to receive notification of that event then it can deregister its interest in the event using the DELETE interface.

Input parameters

EVENT is the event which the caller wishes to deregister its interest in. It can have any of these values:

SMSVSAM_OPERATIONAL

LISTEN_GATE is the gate number of the gate which the caller specified as its listen gate when it registered an interest in this event.

Output parameters

RESPONSE is DFHDMEN's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when response is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LISTEN_NOT_ACTIVE

DMEN gate, NOTIFY_SMSVSAM_OPERATIONAL function

Domains that have registered their interest in ENF events are invoked at their identified listen gates when the ENF event occurs. A unique DMEN notify function is provided for each event to allow event specific parameters to be specified in a meaningful way.

The NOTIFY_SMSVSAM_OPERATIONAL function of the DMEN gate is used to notify domains which have registered an interest in it of the occurrence of the SMSVSAM operational event.

Input parameters

NOTIFY_PLIST is a parameter list specific to the ENF event being notified, which was supplied by the subsystem issuing the ENF signal.

Output parameters

RESPONSE is DFHDMEN's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when response is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	RESTART_RLS_FAILED

DMDM gate, ADD_DOMAIN function

The ADD_DOMAIN function of the DMDM gate adds a new domain to the DM table (on the CICS catalog) of all domains. Because the add is placed on the catalog, it survives system failure. A delete is required to remove the entry.

Input parameters

DOMAIN_NAME is a unique string, 1 through 8 characters, which is the name of the domain.

PROGRAM_NAME is a unique string, 1 through 8 characters, which is the name of the initialization module for the specified domain.

DOMAIN_TOKEN is the unique index that corresponds to the new table entry for the domain.

DOMAIN_ID is the unique character pair, usually an abbreviated form of the domain name.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOADER_ERROR, ABEND, LOOP
EXCEPTION	DUPLICATE_DOMAIN_NAME PROGRAM_NOT_FOUND INSUFFICIENT_STORAGE DUPLICATE_DOMAIN_TOKEN

DMDM gate, QUIESCE_SYSTEM function

The QUIESCE_SYSTEM function of the DMDM gate is used to call the domain manager to cause a normal shutdown of the system.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|INVALID|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	INSUFFICIENT_STORAGE ABEND LOOP
INVALID	SYSTEM_INITIALIZING

DMDM gate, SET_PHASE function

When a domain issues SET_PHASE during initialization, it is declaring that it is now prepared to support a given set of services.

When a domain issues SET_PHASE during quiesce, it is asserting that it still needs the set of services identified by that phase number.

The system phase is the minimum of all active domains' phases during initialization, and the maximum during quiesce.

Input parameters

PHASE specifies the set of services that are to be available.

STATUS is either ACTIVE or INACTIVE.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|INVALID|KERNERROR|DISASTER`

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	SYSTEM_NOT_INITIALIZING SYSTEM_NOT_QUIESCING INVALID_PHASE

DMDM gate, WAIT_PHASE function

The WAIT_PHASE function of the DMDM gate is used to wait until the services required to carry on the work are available.

A WAIT_PHASE for a given phase is understood by CICS as a SET_PHASE for at least the phase specified in the phase parameter of WAIT_PHASE.

Input parameters

PHASE specifies the set of services that are to be available.

STATUS specifies the required status. It is either ACTIVE or INACTIVE.

[DOMAIN_TOKEN]
specifies the domain. If this is omitted, a wait on the system phase is actioned, rather than for a particular domain.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|KERNERROR|DISASTER`

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	DOMAIN_TOKEN_NOT_ACTIVE
INVALID	SYSTEM_NOT_INITIALIZING SYSTEM_NOT_QUIESCING INVALID_PHASE

DMIQ gate, START_BROWSE function

The START_BROWSE function of the DMIQ gate is used to create a browse thread. The GET_NEXT function request issued after this command returns the first domain in the active domain list.

Input parameters

None.

Output parameters

BROWSE_TOKEN is the token identifying this browse session.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|INVALID|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

DMIQ gate, GET_NEXT function

The GET_NEXT function of the DMIQ gate is used to return the next available record or an END indication.

Input parameters

BROWSE_TOKEN is the token identifying this browse session.

Output parameters

DOMAIN_NAME is a unique string, 1 through 8 characters, which is the name of the domain.

PROGRAM_NAME is a unique string, 1 through 8 characters, which is the name of the initialization module for the specified domain.

DOMAIN_TOKEN is the unique index that corresponds to the new table entry for the domain.

DOMAIN_ID is the unique character pair, usually an abbreviated form of the domain name.

DOMAIN_STATUS
is ACTIVE or INACTIVE.

DOMAIN_PHASE is the current phase level for that domain.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	END_LIST
INVALID	BROWSE_TOKEN_NOT_FOUND

DMIQ gate, END_BROWSE function

The END_BROWSE function of the DMIQ gate is used to release the browse thread at any time.

Input parameters

BROWSE_TOKEN is the token identifying this browse session.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|INVALID|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	BROWSE_TOKEN_NOT_FOUND

DMIQ gate, INQ_DOMAIN_BY_NAME function

The INQ_DOMAIN_BY_NAME function of the DMIQ gate is used to get the domain's token, ID, status, and phase for the specified domain name.

Input parameters

DOMAIN_NAME is the unique name of an existing domain.

Output parameters

DOMAIN_TOKEN is the unique index that corresponds to the table entry for the domain.
DOMAIN_ID is the unique character pair, usually an abbreviated form of the domain name.
DOMAIN_STATUS
is ACTIVE or INACTIVE.
DOMAIN_PHASE is the current phase level for that domain.
RESPONSE is the domain's response to the call. It can have any of these values:
OK|INVALID|KERNERROR|DISASTER
[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	DOMAIN_NAME_NOT_FOUND

DMIQ gate, INQ_DOMAIN_BY_TOKEN function

The INQ_DOMAIN_BY_TOKEN function of the DMIQ gate is used to get the domain's name, ID, status, and phase for the specified domain token.

Input parameters

DOMAIN_TOKEN is the unique index that corresponds to the table entry for the domain.

Output parameters

DOMAIN_NAME is a unique string, 1 through 8 characters, which is the name of the domain.
DOMAIN_ID is the unique character pair, usually an abbreviated form of the domain name.
DOMAIN_STATUS
is ACTIVE or INACTIVE.
DOMAIN_PHASE is the current phase level for that domain.
RESPONSE is the domain's response to the call. It can have any of these values:
OK|INVALID|KERNERROR|DISASTER
[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	DOMAIN_TOKEN_NOT_FOUND

DMIQ gate, INQ_DOMAIN_BY_ID function

The INQ_DOMAIN_BY_ID function of the DMIQ gate is used to get the domain's token, name, status, and phase for the specified domain ID.

Input parameters

DOMAIN_ID is the unique character pair, usually an abbreviated form of the domain name.

Output parameters

DOMAIN_TOKEN is the unique index that corresponds to the table entry for the domain.

DOMAIN_NAME is a unique string, 1 through 8 characters, which is the name of the domain.

DOMAIN_STATUS
is ACTIVE or INACTIVE.

DOMAIN_PHASE is the current phase level for that domain.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|INVALID|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	DOMAIN_ID_NOT_FOUND

Domain manager domain's generic gates

Table 42 summarizes the domain manager domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 42. Domain manager domain's generic gates

Gate	Trace	Function	Format
DSAT	None	TASK_REPLY	DSAT

For descriptions of the DSAT function and its input and output parameters, refer to the section dealing with the corresponding generic format:

Functions and parameters

Format DSAT—"Dispatcher domain's generic formats" on page 717

Domain manager domain's generic formats

Table 43 describes the generic formats owned by the domain manager domain and shows the functions performed on the calls.

Table 43. Generic formats owned by the domain manager domain

Format	Calling module	Function
DMDM	DFHKETCB DFHDMDS DFHDMDS DFHKETCB	PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN

In the descriptions of the formats that follow, the "input" parameters are input not to the domain manager domain, but to the domain being called by the domain manager. Similarly, the "output" parameters are output by the domain that was called by the domain manager, in response to the call.

DMDM format, PRE_INITIALIZE function

The DFHKETCB module issues a preinitialization call to each of the following domains: LC, PA, TR, ME, DU, LM, SM, DD, DS, XM, LD, and DM.

Apart from the LD, and DM domains, preinitialization takes place under the job-step TCB; for LD, and DM, it takes place under the resource-owning (RO) TCB.

In preinitialization processing, the domain manager domain reads information about domains from the local catalog, and passes it to the kernel. It then attaches the initialization tasks for all the other domains.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|INVALID|KERNERROR|DISASTER`

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	INSUFFICIENT_STORAGE ABEND LOOP

DMDM format, INITIALIZE_DOMAIN function

The domain manager domain issues an INITIALIZE_DOMAIN function call to a domain. In initialization processing, the domain manager domain performs only internal routines.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|INVALID|KERNERROR|DISASTER`

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	INSUFFICIENT_STORAGE ABEND LOOP
INVALID	ALREADY_INITIALIZED

DMDM format, QUIESCE_DOMAIN function

The domain manager domain issues a QUIESCE_DOMAIN function call to a domain when the system is required to shut down normally. The domain manager domain initiates quiesce processing by attaching the quiesce task for each domain.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|INVALID|KERNERROR|DISASTER`

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	INSUFFICIENT_STORAGE ABEND LOOP

DMDM format, TERMINATE_DOMAIN function

The domain manager domain issues a TERMINATE_DOMAIN function call to a domain when the system is required to shut down quickly. This call is always issued under the job-step TCB.

The domain manager domain does no termination processing.

Input parameters

CLEAN_UP
indicates whether or not the TERMINATE_DOMAIN function request is being issued under a cleanup-only ESTAE exit. It can have either of these values:

`YES|NO`

YES implies restrictions for termination logic, specifically that an ATTACH request cannot be issued.

CANCEL
indicates whether or not the termination is happening because of an operator CANCEL command. It can have either of these values:

`YES|NO`

YES means that attached subtasks are no longer dispatchable.

TERMINATION_TYPE
indicates whether the termination is happening because of either a quiesce or an abnormal shutdown. It can have either of these values:

`QUIESCE|IMMEDIATE`

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|INVALID|KERNERROR|DISASTER`

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

Modules

Module	Function
DFHDMDM	Handles the following requests: INITIALIZE_DOMAIN PRE_INITIALIZE QUIESCE_DOMAIN QUIESCE_SYSTEM TERMINATE_DOMAIN SET_PHASE WAIT_PHASE ADD_DOMAIN
DFHDMDS	Handles the TASK_REPLY request
DFHDMDUF	Formats the DM domain control blocks in a CICS system dump
DFHDMEN	Handles LISTEN, DELETE, NOTIFY_SMSVSAM_OPERATIONAL

Module	Function
DFHDMENF	Broadcasts ENF events to interested domains
DFHDMIQ	Handles the following requests: START_BROWSE GET_NEXT END_BROWSE INQUIRE_DOMAIN_BY_ID INQUIRE_DOMAIN_BY_NAME INQUIRE_DOMAIN_BY_TOKEN
DFHDMSVC	Provides authorized services for the DM ENF support
DFHDMTRI	Interprets DM domain trace entries
DFHDMWQ	Handles the following requests: INITIALIZE SET_UP_WAIT RESUME_WAITERS RESUME_DOMAIN_WAITERS RESUME_PHASE_WAITERS

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the domain manager domain are of the form DM xxxx; the corresponding trace levels are DM 1, DM 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 79. Debugging profile domain (DP)

The Debugging profile domain manages debugging profiles.

Debugging profile domain's specific gates

Table 44 summarizes the Debugging profile domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, and the functions provided by the gates.

Table 44. Debugging profile domain's specific gates

Gate	Trace	Function	XPI
DPFM	DP 0300 DP 0301	GET_DEBUG_PROFILE SAVE_DEBUG_PROFILE DELETE_DEBUG_PROFILE ACTIVATE_DEBUG_PROFILE INACTIVATE_DEBUG_PROFILE REPLACE_DEBUG_PROFILE START_PM_BROWSE READNEXT_PM_PROFILE END_PM_BROWSE	No
DPIQ	DP 0A00 DP 0A01	INQUIRE_PARAMETERS SET_PARAMETERS INQUIRE_DEBUG_TASK SET_DEBUG_PROFILE	No
DPLM	DP 0200 DP 0201	STARTBR_DEBUG_PROFILES READNEXT_DEBUG_PROFILE READNEXT_INPUT ENDBR_DEBUG_PROFILES RESTARTBR_DEBUG_PROFILES UPDATE_PROFILE_IN_LIST	No
DPPM	DP 0800 DP 0801	PATTERN_MATCH_TASK PATTERN_MATCH_PROFILE	No
DPUM	DP 0500 DP 0501	GET_USER_DEFAULTS SAVE_USER_DEFAULTS	No
DPWD	DP 1100 DP 1101	PROCESS_PAGE PROCESS_SUBMIT	No
DPWE	DP 0F00 DP 0F01	PROCESS_PAGE PROCESS_SUBMIT	No
DPWJ	DP 1000 DP 1001	PROCESS_PAGE PROCESS_SUBMIT	No
DPWL	DP 0E00 DP 0E01	PROCESS_PAGE PROCESS_SUBMIT	No
DPXM	DP 0900 DP0901	INIT_XM_CLIENT BIND_XM_CLIENT RELEASE_XM_CLIENT	No

DPFM gate, GET_DEBUG_PROFILE function

Retrieve a debugging profile from the debugging profile data set.

Input parameters

OWNER_USERID The userid of the debugging profile's owner

PROFILE_NAME The name of the debugging profile

[BEAN_BLOCK] A block of storage containing the bean name

[CLASS_BLOCK]
A block of storage containing the class name

[METHOD_BLOCK]
A block of storage containing the method name

[IP_NAME_OR_ADDR_BLOCK]
A block of storage containing the IP name or IP address

[LE_OPTIONS_BLOCK]
A block of storage containing Language Environment options

Output parameters

[TRANID] The transaction ID specified in the debugging profile
[TERMID] The terminal ID specified in the debugging profile
[PROGRAM] The program name specified in the debugging profile
[COMP_UNIT] The compile unit name specified in the debugging profile
[TYPE] The type of debugging profile. Values are `C` | `E` | `J` | `LE`
[USERID] The user ID specified in the debugging profile
[NETNAME] The terminal's network name specified in the debugging profile
[APPLID] The Applid specified in the debugging profile
[SESSION_TYPE]
The session type specified in the debugging profile. Values are `LU3270` | `TCP`
[PORT] The port number specified in the debugging profile
[LU_3270_DISPLAY]
The 3270 display terminal to be used by Debug Tool
[JVM_PROFILE]
The JVM profile specified in the debugging profile
[TEST_LEVEL] The test level specified in the debugging profile. Values are `ALL` | `ERROR` | `NONE`
[COMMAND_FILE]
The command file specified in the debugging profile
[PROMPT] The prompt specified in the debugging profile
[PREFERENCE_FILE]
The preference file specified in the debugging profile
[STATUS] The status of the debugging profile. Values are `ACTIVE` | `INACTIVE`
[PATTERN_MATCH_NUMBER]
A metric computed from the contents of the debugging profile, which is compared with the pattern match number form other profiles to determine which of the profiles is the best match for a program instance.
[ACTIVATE_USERID]
For an active debugging profile, the user ID of the user who made it active.
RESPONSE The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`
[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	PROFILE_NOT_FOUND FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPFM gate, SAVE_DEBUG_PROFILE function

Save a debug profile on the debug profile data set.

Input parameters

OWNER_USERID The userid of the profile's owner
PROFILE_NAME The name of the debugging profile
[TRANID] The transaction ID specified in the debugging profile
[TERMID] The terminal ID specified in the debugging profile
[PROGRAM] The program name specified in the debugging profile
[COMP_UNIT] The compile unit name specified in the debugging profile
[TYPE] The type of debugging profile. Values are `C` | `E` | `J` | `LE`
[BEAN_BLOCK] A block of storage containing the bean name

[CLASS_BLOCK]
A block of storage containing the class name

[METHOD_BLOCK]
A block of storage containing the method name

[USERID] The user ID specified in the debugging profile

[NETNAME] The terminal's network name specified in the debugging profile

[APPLID] The Applid specified in the debugging profile

SESSION_TYPE The session type specified in the debugging profile. Values are `LU3270` | `TCP`

[IP_NAME_OR_ADDR_BLOCK]
A block of storage containing the IP name or IP address

[PORT] The port number specified in the debugging profile

[LU_3270_DISPLAY]
The 3270 display terminal specified in the debugging profile to be used by Debug Tool

[JVM_PROFILE]
The JVM profile specified in the debugging profile

[TEST_LEVEL] The test level specified in the debugging profile. Values are `ALL` | `ERROR` | `NONE`

[COMMAND_FILE]
The command file specified in the debugging profile

[PROMPT] The prompt specified in the debugging profile

[PREFERENCE_FILE]
The preference file specified in the debugging profile

[LE_OPTIONS_BLOCK]
A block of storage containing Language Environment options

Output parameters

MANGLE_CODE Values are `PROPERTY_ACC` | `UNDERSCORE` | `IDL_KEYWORD` | `MANGLED_TO_SELF`

RESPONSE The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	DUPLICATE_PROFILE PROFILE_NAME_BLANK PROFILE_NAME_INVALID TRANID_INVALID TERMID_INVALID PROGRAM_INVALID COMP_UNIT_INVALID USERID_INVALID NETNAME_INVALID APPLID_INVALID JVM_PROFILE_INVALID CMD_FILE_INVALID PREF_FILE_INVALID CLASS_INVAL_FOR_TYPE_E BEAN_INVAL_FOR_TYPE_J BEAN_INVAL_FOR_TYPE_C METHOD_INVAL_FOR_TYPE_J CLASS_INVALID BEAN_INVALID METHOD_INVALID PROMPT_INVALID FILE_ERROR FILE_FULL
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPFM gate, DELETE_DEBUG_PROFILE function

Delete a debugging profile from the debugging profile data set.

Input parameters

CURRENT_USERID
The userid of the user making the request

OWNER_USERID
The userid of the profile's owner

PROFILE_NAME
The name of the debugging profile

Output parameters

RESPONSE
The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	PROFILE_NOT_FOUND PROFILE_ACTIVE FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPFM gate, ACTIVATE_DEBUG_PROFILE function

Input parameters

CURRENT_USERID
The userid of the user making the request

OWNER_USERID
The userid of the profile's owner

PROFILE_NAME
The name of the debugging profile

[SESSION_TYPE]
The session type specified in the debugging profile. Values are LU3270 | TCP

[IP_NAME_OR_ADDR_BLOCK]
A block of storage containing the IP name or IP address

[PORT]
The port number specified in the debugging profile

[LU_3270_DISPLAY]
The 3270 display terminal specified in the debugging profile to be used by Debug Tool

Output parameters

[PATTERN_MATCH_NUMBER]
A metric computed from the contents of the debugging profile, which is compared with the pattern match number form other profiles to determine which of the profiles is the best match for a program instance.

RESPONSE
The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	PROFILE_NOT_FOUND ALREADY_ACTIVE FILE_ERROR FILE_FULL
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPFM gate, INACTIVATE_DEBUG_PROFILE function

Inactivate a debug_profile on the debugging profile data set.

Input parameters

CURRENT_USERID
The userid of the user making the request

OWNER_USERID The userid of the profile's owner

PROFILE_NAME The name of the debugging profile

Output parameters

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	PROFILE_NOT_FOUND ALREADY_INACTIVE FILE_ERROR FILE_FULL
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPFM gate, REPLACE_DEBUG_PROFILE function

Replace a debug_profile on the debugging profile data set.

Input parameters

OWNER_USERID The userid of the profile's owner

PROFILE_NAME The name of the debugging profile

[TRANID] The transaction ID specified in the debugging profile

[TERMID] The terminal ID specified in the debugging profile

[PROGRAM] The program name specified in the debugging profile

[COMP_UNIT] The compile unit name specified in the debugging profile

[TYPE] The type of debugging profile. Values are C | E | J | LE

[BEAN_BLOCK] A block of storage containing the bean name

[CLASS_BLOCK]
A block of storage containing the class name

[METHOD_BLOCK]
A block of storage containing the method name

[USERID] The user ID specified in the debugging profile

[NETNAME] The terminal's network name specified in the debugging profile

[APPLID] The Applid specified in the debugging profile

[SESSION_TYPE]
The session type specified in the debugging profile. Values are LU3270 | TCP

[IP_NAME_OR_ADDR_BLOCK]
A block of storage containing the IP name or IP address

[PORT] The port number specified in the debugging profile

[LU_3270_DISPLAY]
The 3270 display terminal specified in the debugging profile to be used by Debug Tool

[JVM_PROFILE]
The JVM profile specified in the debugging profile

[TEST_LEVEL] The test level specified in the debugging profile. Values are ALL | ERROR | NONE

[COMMAND_FILE]
The command file specified in the debugging profile

[PROMPT] The prompt specified in the debugging profile

[PREFERENCE_FILE]
The preference file specified in the debugging profile

[LE_OPTIONS_BLOCK]
A block of storage containing Language Environment options

Output parameters

NEW_PROFILE_CREATED
Indicates whether a new profile was created. Values are YES | NO

[MANGLE_CODE]
Values are PROPERTY_ACC | UNDERSCORE | IDL_KEYWORD | MANGLED_TO_SELF

RESPONSE
The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	PROFILE_NAME_BLANK PROFILE_NAME_INVALID TRANID_INVALID TERMID_INVALID PROGRAM_INVALID COMP_UNIT_INVALID USERID_INVALID NETNAME_INVALID APPLID_INVALID JVM_PROFILE_INVALID CMD_FILE_INVALID PREF_FILE_INVALID CLASS_INVAL_FOR_TYPE_E BEAN_INVAL_FOR_TYPE_J BEAN_INVAL_FOR_TYPE_C METHOD_INVAL_FOR_TYPE_J CLASS_INVALID BEAN_INVALID METHOD_INVALID PROMPT_INVALID FILE_ERROR FILE_FULL
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPFM gate, START_PM_BROWSE function

Start a browse for pattern matching.

Input parameters

[MATCH_TYPE]
Values are TYPE_LE | TYPE_J

Output parameters

RESPONSE
The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	NO_PROFILES FILE_ERROR

RESPONSE	REASON
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION
INVALID	
KERNERROR	
PURGED	

DPFM gate, READNEXT_PM_PROFILE function

Read the next profile on the debugging profile data set for pattern match.

Input parameters

[BEAN_BLOCK] A block of storage containing the bean name

[CLASS_BLOCK]
A block of storage containing the class name

[MANGLED_METHOD_BLOCK]
A block of storage containing the mangled method name

[IP_NAME_OR_ADDR_BLOCK]
A block of storage containing the IP name or IP address

[LE_OPTIONS_BLOCK]
A block of storage containing Language Environment options

Output parameters

OWNER_USERID The userid of the profile's owner

PROFILE_NAME The name of the debugging profile

TRANID The transaction ID specified in the debugging profile

TERMID The terminal ID specified in the debugging profile

PROGRAM The program name specified in the debugging profile

[COMP_UNIT] The compile unit name specified in the debugging profile

TYPE The type of debugging profile. Values are C | E | J | LE

USERID The user ID specified in the debugging profile

NETNAME The terminal's network name specified in the debugging profile

APPLID The Applid specified in the debugging profile

SESSION_TYPE The session type specified in the debugging profile. Values are LU3270 | TCP

PORT The port number specified in the debugging profile

LU_3270_DISPLAY
The 3270 display terminal specified in the debugging profile to be used by Debug Tool

JVM_PROFILE The JVM profile specified in the debugging profile

TEST_LEVEL The test level specified in the debugging profile. Values are ALL | ERROR | NONE

COMMAND_FILE The command file specified in the debugging profile

PROMPT The prompt specified in the debugging profile

PREFERENCE_FILE
The preference file specified in the debugging profile

PATTERN_MATCH_NUMBER
A metric computed from the contents of the debugging profile, which is compared with the pattern match number form other profiles to determine which of the profiles is the best match for a program instance.

[ACTIVATE_USERID]
For an active debugging profile, the user ID of the user who made it active.

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	END_OF_PROFILES FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPFM gate, END_PM_BROWSE function

End the browse for pattern matching.

Input parameters

None

Output parameters

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPIQ gate, INQUIRE_PARAMETERS function

Inquire DP domain parameters.

Input parameters

None

Output parameters

[DEBUGTOOL] The value of the DEBUGTOOL system initialization parameter. Values are DEBUGTOOL_YES | DEBUGTOOL_NO

[DTLEVEL] Specifies whether the level of Debug Tool supports the CADP transaction. Values are DTNEW_YES | DTNEW_NO

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

DPIQ gate, SET_PARAMETERS function

Set DP domain parameters.

Input parameters

DEBUGTOOL The value of the DEBUGTOOL system initialization parameter. Values are DEBUGTOOL_YES | DEBUGTOOL_NO

Output parameters

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

DPIQ gate, INQUIRE_DEBUG_TASK function

Inquire DP domain debug settings.

Input parameters

None

Output parameters

[DEBUG_TASK] Specifies whether Debug Tool is to be used to debug an application. Values are YES | NO

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

DPIQ gate, SET_DEBUG_PROFILE function

Set DP domain parameters.

Input parameters

[DEBUG_PROFILE]
Values are YES | NO

Output parameters

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

DPLM gate, STARTBR_DEBUG_PROFILES function

Creates an in-memory linked list copy of the debug profiles from the debug profile data set. The list is filtered and sorted as requested. A browse_list_token which uniquely identifies the list on subsequent requests to process it is returned.

Input parameters

CURRENT_USERID
The userid of the user making the request

FILTER_ACTIVE
Specifies whether the list contains active profiles only, or active and inactive profiles. Values are ACTIVE_P | ALL_P

FILTER_USER Specifies whether the list contains profiles for just the current user, or all users. Values are CURRENT_USER | ALL_U

SORT_TYPE Specifies the field used to sort the list. Values are OWNER | NAME | TRAN | STAT | PROG | TERM | USER | APPL | NETN | COMP_U | TYP

Output parameters

BROWSE_LIST_TOKEN
A token which uniquely identifies the list of profiles.

[NUMBER_IN_LIST]
The number of profiles in the list

[CURRENT_PAGE]
Specifies which page of the list of profiles is currently displayed

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	NO_PROFILES FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPLM gate, READNEXT_DEBUG_PROFILE function

Returns one profile to the caller for display on the screen. Largely for the benefit of the 3270 version of CADP, the readnext can optionally position itself based on a page size parameter so that it is possible to easily implement scrolling up and down. The default if no position is specified is to return the next profile.

Input parameters

BROWSE_LIST_TOKEN
A token which uniquely identifies the list of profiles.

[PAGE_SIZE] The number of profiles which can be shown on a page of the display

[POSITION] Specifies the position in the list of the next profile to be read. Values are `TOP` | `TOP_CURRENT_PAGE` | `NEXT_PROFILE` | `PAGE_FORWARD` | `PAGE_BACK`

[BEAN_BLOCK] A block of storage containing the bean name

[CLASS_BLOCK]
A block of storage containing the class name

[METHOD_BLOCK]
A block of storage containing the method name

[MANGLED_METHOD_BLOCK]
A block of storage containing the mangled method name

[LE_OPTIONS_BLOCK]
A block of storage containing Language Environment options

Output parameters

INPUT The action specified for the profile. Values are `ACTIVATE` | `INACTIVATE` | `COPY` | `DELETE` | `CLEAR`

OWNER_USERID The userid of the profile's owner

PROFILE_NAME The name of the debugging profile

TRANID The transaction ID specified in the debugging profile

TERMID The terminal ID specified in the debugging profile

PROGRAM The program name specified in the debugging profile

COMP_UNIT The compile unit name specified in the debugging profile

TYPE The type of debugging profile. Values are `C` | `E` | `J` | `LE`

USERID The user ID specified in the debugging profile

NETNAME The terminal's network name specified in the debugging profile

APPLID The Applid specified in the debugging profile

JVM_PROFILE The JVM profile specified in the debugging profile

TEST_LEVEL The test level specified in the debugging profile. Values are `ALL` | `ERROR` | `NONE`

COMMAND_FILE The command file specified in the debugging profile

PROMPT The prompt specified in the debugging profile

PREFERENCE_FILE
The preference file specified in the debugging profile

STATUS The status of the debugging profile. Values are `ACTIVE` | `INACTIVE`

PATTERN_MATCH_NUMBER
A metric computed from the contents of the debugging profile, which is compared with the pattern match number form other profiles to determine which of the profiles is the best match for a program instance.

[PROFILE_NUMBER]
The position of the current profile in the list

[CURRENT_PAGE]
Specifies which page of the list of profiles is currently displayed

[ACTIVATE_USERID]
For an active debugging profile, the user ID of the user who made it active.

[INVALID_INPUT]
Whatever was (invalidly) typed as an input

RESPONSE The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	END_OF_PROFILES ALREADY_AT_TOP ALREADY_AT_BOTTOM
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPLM gate, READNEXT_INPUT function

When inputs are typed in against profiles they are saved with the profile in the linked list so that they are still retrievable for redisplay after scrolling up and down. Readnext_input allows easy retrieval of just those profiles with inputs against them so that they can be processed when enter is pressed. All the data in the profile is returned as it is required if the input to be processed is COPY.

Input parameters

BROWSE_LIST_TOKEN
A token which uniquely identifies the list of profiles.

[POSITION] Specifies the position in the list of the next profile to be read. Values are TOP | NEXT_PROFILE

[INPUT_FILTER]
Specifies profiles of interest, based on any actions that have been specified for the profile. Values are ACTIVATES | INACTIVATES | COPIES | DELETES | ALL_INPUTS

[BEAN_BLOCK] A block of storage containing the bean name

[CLASS_BLOCK]
A block of storage containing the class name

[METHOD_BLOCK]
A block of storage containing the method name

[MANGLED_METHOD_BLOCK]
A block of storage containing the mangled method name

[LE_OPTIONS_BLOCK]
A block of storage containing Language Environment options

Output parameters

INPUT The action specified for the profile. Values are ACTIVATE | INACTIVATE | COPY | DELETE | CLEAR

OWNER_USERID The userid of the profile's owner

PROFILE_NAME The name of the debugging profile

TRANID The transaction ID specified in the debugging profile

TERMID The terminal ID specified in the debugging profile

PROGRAM The program name specified in the debugging profile

COMP_UNIT The compile unit name specified in the debugging profile

TYPE The type of debugging profile. Values are C | E | J | LE

USERID The user ID specified in the debugging profile

NETNAME The terminal's network name specified in the debugging profile

APPLID The Applid specified in the debugging profile

JVM_PROFILE The JVM profile specified in the debugging profile

TEST_LEVEL The test level specified in the debugging profile. Values are ALL | ERROR | NONE

COMMAND_FILE The command file specified in the debugging profile

PROMPT The prompt specified in the debugging profile

PREFERENCE_FILE
The preference file specified in the debugging profile

STATUS The status of the debugging profile. Values are ACTIVE | INACTIVE

[PATTERN_MATCH_NUMBER]
A metric computed from the contents of the debugging profile, which is compared with the pattern match number form other profiles to determine which of the profiles is the best match for a program instance.

[ACTIVATE_USERID]
For an active debugging profile, the user ID of the user who made it active.

[INVALID_INPUT]
Whatever was (invalidly) typed as an input

RESPONSE
The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	END_OF_INPUTS
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION
INVALID	
KERNERROR	
PURGED	

DPLM gate, ENDBR_DEBUG_PROFILES function

Frees the in-memory list of debug profile records.

Input parameters

BROWSE_LIST_TOKEN
A token which uniquely identifies the list of profiles.

Output parameters

RESPONSE
The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPLM gate, RESTARTBR_DEBUG_PROFILES function

A new in_memory list of the debug profiles is created from the profiles on the data set but the previous filter, sort and position in the list currently displayed are maintained. The position may alter if the underlying list has changed dramatically (due to additions/deletions of profiles via other CADP transactions). Outstanding actions are preserved, and copied to the new list.

Input parameters

BROWSE_LIST_TOKEN
A token which uniquely identifies the list of profiles.

CURRENT_USERID
The userid of the user making the request

Output parameters

[NUMBER_IN_LIST]
The number of profiles in the list

[CURRENT_PAGE]
Specifies which page of the list of profiles is currently displayed

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	NO_PROFILES FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPLM gate, UPDATE_PROFILE_IN_LIST function

Update the specified in-memory linked list element with the input supplied so that it may be kept until ready to process later. CLEAR may be used to clear an input that has been handled.

Input parameters

BROWSE_LIST_TOKEN
A token which uniquely identifies the list of profiles.

OWNER_USERID The userid of the profile's owner

PROFILE_NAME The name of the debugging profile

INPUT The action specified for the profile. Values are ACTIVATE | INACTIVATE | COPY | DELETE | CLEAR

Output parameters

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	REASON
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPPM gate, PATTERN_MATCH_TASK function

Determines if an active debugging profile matches the parameters supplied.

Input parameters

TRANID The transaction ID to be matched with the corresponding field in the active debugging profiles

TERMID The terminal ID to be matched with the corresponding field in the active debugging profiles

USERID The user ID to be matched with the corresponding field in the active debugging profiles

NETNAME The terminal's network name to be matched with the corresponding field in the active debugging profiles

APPLID The APPLID to be matched with the corresponding field in the active debugging profiles

Output parameters

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	NO_MATCH FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPPM gate, PATTERN_MATCH_PROFILE function

Determines if an active debugging profile matches the parameters supplied.

Input parameters

[MATCH_TYPE] Values are LE | NON_LE

[TRANID] The transaction ID to be matched with the corresponding field in the active debugging profiles

[TERMID] The terminal ID to be matched with the corresponding field in the active debugging profiles

[PROGRAM] The program name to be matched with the corresponding field in the active debugging profiles

[COMP_UNIT] The compile unit name to be matched with the corresponding field in the active debugging profiles

[CLASS_BLOCK]
A block of storage containing the class name to be matched with the corresponding field in the active debugging profiles

[MANGLED_METHOD_BLOCK]
A block of storage containing the mangled method name to be matched with the corresponding field in the active debugging profiles

[IP_NAME_OR_ADDR_BLOCK]
A block of storage containing the IP name or IP address

[LE_OPTIONS_BLOCK]
A block of storage containing Language Environment options

[USERID] The user ID to be matched with the corresponding field in the active debugging profiles

[NETNAME] The terminal's network name to be matched with the corresponding field in the active debugging profiles

[APPLID] The Applid to be matched with the corresponding field in the active debugging profiles

Output parameters

[PROFILE_TRANID]
The transaction ID specified in the matching profile

[PROFILE_TERMID]
The terminal ID specified in the matching profile

[PROFILE_PROGRAM]
The program name specified in the matching profile

[PROFILE_COMP_UNIT]
The compile unit name specified in the matching profile

[PROFILE_USERID]
The user ID specified in the matching profile

[PROFILE_NETNAME]
The terminal's network name specified in the matching profile

[PROFILE_APPLID]
The Applid specified in the matching profile

[SESSION_TYPE]
The session type specified in the debugging profile. Values are LU3270 | TCP

[PORT] The port number specified in the debugging profile

[LU_3270_DISPLAY]
The 3270 display terminal specified in the debugging profile to be used by Debug Tool

[JVM_PROFILE]
The JVM profile specified in the debugging profile

[TEST_LEVEL]
The test level specified in the debugging profile. Values are ALL | ERROR | NONE

[COMMAND_FILE]
The command file specified in the debugging profile

[PROMPT]
The prompt specified in the debugging profile

[PREFERENCE_FILE]
The preference file specified in the debugging profile

RESPONSE
The domain's response to the call. Values are OK | EXCEPTION | DISASTER | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	NO_MATCH FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPUM gate, GET_USER_DEFAULTS function

Get user defaults. If none already, returns global defaults.

Input parameters

CURRENT_SESSION_TYPE
The session type specified for the current user. Values are LU3270 | TCP

CURRENT_USERID
The userid of the user making the request

[IP_NAME_OR_ADDR_BLOCK]
A block of storage containing the IP name or IP address

[LE_OPTIONS_BLOCK]
A block of storage containing Language Environment options

Output parameters

[SUPPRESS_PANEL]
Specifies whether the debugging device panel is to be suppressed. Values are SUPPRESS | NOSUPPRESS

[SESSION_TYPE]
The session type specified in the debugging profile. Values are LU3270 | TCP

[PORT]
The port number specified in the debugging profile

[LU_3270_DISPLAY]
The 3270 display terminal specified in the debugging profile to be used by Debug Tool

[JVM_PROFILE]
The JVM profile specified in the debugging profile

[TEST_LEVEL]
The test level specified in the debugging profile. Values are ALL | ERROR | NONE

[COMMAND_FILE]
The command file specified in the debugging profile

[PROMPT]
The prompt specified in the debugging profile

[PREFERENCE_FILE]
The preference file specified in the debugging profile

[FILTER_ACTIVE]
Specifies whether the list contains active profiles only, or active and inactive profiles. Values are ACTIVE_P | ALL_P

[FILTER_USER]
Specifies whether the list contains profiles for just the current user, or all users. Values are CURRENT_USER | ALL_U

[SORT_TYPE] Specifies the field used to sort the list. Values are OWNER | NAME | TRAN | STAT | PROG | TERM | USER | APPL | NETN | COMP_U | TYP

[TYPE] The type of debugging profile. Values are C | E | J | LE

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPUM gate, SAVE_USER_DEFAULTS function

Save user defaults. Never returns duplicate response - saves or updates.

Input parameters

CURRENT_USERID
The userid of the user making the request

[SUPPRESS_PANEL]
Specifies whether the debugging device panel is to be suppressed. Values are SUPPRESS | NOSUPPRESS

[SESSION_TYPE]
The session type specified in the debugging profile. Values are LU3270 | TCP

[IP_NAME_OR_ADDR_BLOCK]
A block of storage containing the IP name or IP address

[PORT] The port number specified in the debugging profile

[LU_3270_DISPLAY]
The 3270 display terminal specified in the debugging profile to be used by Debug Tool

[JVM_PROFILE]
The JVM profile specified in the debugging profile

[TEST_LEVEL] The test level specified in the debugging profile. Values are ALL | ERROR | NONE

[COMMAND_FILE]
The command file specified in the debugging profile

[PROMPT] The prompt specified in the debugging profile

[PREFERENCE_FILE]
The preference file specified in the debugging profile

[LE_OPTIONS_BLOCK]
A block of storage containing Language Environment options

[FILTER_ACTIVE]
Specifies whether the list contains active profiles only, or active and inactive profiles. Values are ACTIVE_P | ALL_P

[FILTER_USER]
Specifies whether the list contains profiles for just the current user, or all users. Values are CURRENT_USER | ALL_U

[SORT_TYPE] Specifies the field used to sort the list. Values are OWNER | NAME | TRAN | STAT | PROG | TERM | USER | APPL | NETN | COMP_U | TYP

[TYPE] The type of debugging profile. Values are C | E | J | LE

Output parameters

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	3270_DISPLAY_BLANK 3270_DISPLAY_INVALID PORT_BLANK PORT_INVALID IP_BLANK IP_INVALID JVM_PROFILE_INVALID CMD_FILE_INVALID PROMPT_INVALID PREF_FILE_INVALID FILE_ERROR FILE_FULL
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPWD gate, PROCESS_PAGE function

Process a request for an html page in the following format:

`http://`*`mvs_address`*`:`*`port`*`/CICS/CWBA/dfhdpwb?`*`options`*

The input options will be read by the page processor from ITOKEN. The page processor will generate an output page request in OTOKEN.

Input parameters

PAGE The page to be processed

ITOKEN A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

[MSG_TYPE] The type of message to be displayed when the page is formatted, in the absence of a more serious message. If this value is not present then by default no message is displayed. Values are `INFO` | `ERROR`

[MSG_NUM] The message number of a message to be displayed when the page is formatted.

[MSG_INSERT1]
An insert for the message. If this field is null there is no first insert.

[MSG_INSERT2]
An insert for the message. If this field is null there is no second insert.

Output parameters

OTOKEN A token representing a chain of output html tags.

RESPONSE The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPWD gate, PROCESS_SUBMIT function

Process a submitted form request. The input options will be read by the page processor from ITOKEN. The page processor will generate an output page request in OTOKEN.

Input parameters

BUTTON
The action button used to submit the form.

ITOKEN
A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

Output parameters

OTOKEN
A token representing a chain of output html tags.

RESPONSE
The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPWE gate, PROCESS_PAGE function

Process a request for an html page in the following format:

`http://`*mvs_address*`:`*port*`/CICS/CWBA/dfhdpwb?`*options*

The input options will be read by the page processor from ITOKEN. The page processor will generate an output page request in OTOKEN.

Input parameters

PAGE
The page to be processed

ITOKEN
A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

[MSG_TYPE]
The type of message to be displayed when the page is formatted, in the absence of a more serious message. If this value is not present then by default no message is displayed. Values are `INFO` | `ERROR`

[MSG_NUM]
The message number of a message to be displayed when the page is formatted.

[MSG_INSERT1]
An insert for the message. If this field is null there is no first insert.

[MSG_INSERT2]
An insert for the message. If this field is null there is no second insert.

Output parameters

OTOKEN
A token representing a chain of output html tags.

RESPONSE
The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPWE gate, PROCESS_SUBMIT function

Process a submitted form request. The input options will be read by the page processor from ITOKEN. The page processor will generate an output page request in OTOKEN.

Input parameters

BUTTON
The action button used to submit the form.

ITOKEN
A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

Output parameters

OTOKEN
A token representing a chain of output html tags.

RESPONSE
The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPWJ gate, PROCESS_PAGE function

Process a request for an html page in the following format:

`http://`*`mvs_address`*`:`*`port`*`/CICS/CWBA/dfhdpwb?`*`options`*

The input options will be read by the page processor from ITOKEN. The page processor will generate an output page request in OTOKEN.

Input parameters

PAGE
The page to be processed

ITOKEN
A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

Output parameters

OTOKEN
A token representing a chain of output html tags.

RESPONSE
The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPWJ gate, PROCESS_SUBMIT function

Process a submitted form request. The input options will be read by the page processor from ITOKEN. The page processor will generate an output page request in OTOKEN.

Input parameters

BUTTON
The action button used to submit the form.

ITOKEN
A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

Output parameters

OTOKEN
A token representing a chain of output html tags.

RESPONSE
The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPWL gate, PROCESS_PAGE function

Process a request for an html page in the following format:

`http://`*`mvs_address`*`:`*`port`*`/CICS/CWBA/dfhdpwb?`*`options`*

The input options will be read by the page processor from ITOKEN. The page processor will generate an output page request in OTOKEN.

Input parameters

PAGE The page to be processed

ITOKEN A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

[MSG_TYPE] The type of message to be displayed when the page is formatted, in the absence of a more serious message. If this value is not present then by default no message is displayed. Values are `INFO` | `ERROR`

[MSG_NUM] The message number of a message to be displayed when the page is formatted.

[MSG_INSERT1]
An insert for the message. If this field is null there is no first insert.

[MSG_INSERT2]
An insert for the message. If this field is null there is no second insert.

Output parameters

OTOKEN A token representing a chain of output html tags.

RESPONSE The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPWL gate, PROCESS_SUBMIT function

Process a submitted form request. The input options will be read by the page processor from ITOKEN. The page processor will generate an output page request in OTOKEN.

Input parameters

BUTTON The action button used to submit the form.

ITOKEN A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

Output parameters

OTOKEN A token representing a chain of output html tags.

RESPONSE The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	FILE_ERROR
DISASTER	ABEND INTERNAL_ERROR DISASTER_PERCOLATION

DPXM gate, INIT_XM_CLIENT function

The INIT_XM_CLIENT call flows from the transaction manager to the DP Domain during transaction initialization. The DP domain allocates the DP domain transaction lifetime control block, and anchors it in the AP domain's transaction token.

Input parameters

[PRINCIPAL_USER_TOKEN]
The token which represents the characteristics of the principal user of the transaction.

Output parameters

RESPONSE The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	INVALID_USER_TOKEN
DISASTER	ABEND LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

DPXM gate, BIND_XM_CLIENT function

The BIND_XM_CLIENT call flows from the transaction manager to the DP Domain during transaction initialization after Recovery Manager initialisation is complete. The DP domain does a scan of the active debugging profiles to determine if it is possible that debugging could be required in this transaction. If it is not then DP domain is not invoked again until transaction termination.

Input parameters

[PRINCIPAL_USER_TOKEN]
The token which represents the characteristics of the principal user of the transaction.

Output parameters

RESPONSE The domain's response to the call. Values are `OK` | `EXCEPTION` | `DISASTER` | `INVALID` | `KERNERROR` | `PURGED`

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	INVALID_USER_TOKEN
DISASTER	ABEND LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

DPXM gate, RELEASE_XM_CLIENT function

The RELEASE_XM_CLIENT call is made from the transaction manager to the DP Domain during transaction termination. DP domain transaction lifetime resources are released.

Input parameters

[PRINCIPAL_USER_TOKEN]
The token which represents the characteristics of the principal user of the transaction.

Output parameters

RESPONSE The domain's response to the call. Values are OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	REASON
EXCEPTION	INVALID_USER_TOKEN
DISASTER	ABEND LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

Debugging profile domain’s generic gates

Table 45 summarizes the Debugging profile domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 45. Debugging profile domain’s generic gates

Gate	Trace	Function	Format
DPDM	DP 0101 DP 0102	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DPDM

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—“Domain manager domain’s generic formats” on page 669

In preinitialization the Debugging profile adds its general subpool and global lock.

In initialization, quiesce, and termination processing, the Debugging profile domain performs only internal routines.

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the Debugging profile domain are of the form DP xxxx; the corresponding trace levels are DP 1, DP 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 80. Dispatcher domain (DS)

The dispatcher domain is concerned with the attaching, running, and detaching of tasks, and the posting of TCBs with the following modes (names): concurrent (CO), ONC/RPC-owning (RP), quasi-reentrant (QR), resource-owning (RO), file-owning (FO), secondary LU usage (SZ), open key 8 (L8), JVM key 8 (J8), JVM key 9 (J9), master JVM (JM), hot-pooling key 8 (H8), sockets (SO), sockets listener (SL), secure sockets key 8 (S8), or DB2 (D2).

Dispatcher domain's specific gates

Table 46 summarizes the dispatcher domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 46. Dispatcher domain's specific gates

Gate	Trace	Function	XPI
DSAT	DS 0002 DS 0003	ATTACH CANCEL_TASK CHANGE_MODE CHANGE_PRIORITY CLEAR_MATCH DELETE_SUBSPACE_TCBS FREE_SUBSPACE_TCBS NOTIFY_DELETE_TCB RELEASE_OPEN_TCB SET_PRIORITY SET_TRANSACTION_TOKEN TCB_POOL_MANAGEMENT	NO NO NO YES NO NO NO NO NO NO NO NO
DSBR	DS 0010 DS 0011	END_BROWSE GET_NEXT INQUIRE_TASK INQUIRE_TCB SET_TASK SET_TCB START_BROWSE	NO NO NO NO NO NO NO
DSIT	DS 0008 DS 0009	ACTIVATE_MODE ADD_TCB DELETE_ALL_OPEN_TCBS DELETE_OPEN_TCB DELETE_TCB FREE_TCB INQUIRE_DISPATCHER PROCESS_DEAD_TCBS SET_DISPATCHER	NO NO NO NO NO NO NO NO NO
DSSR	DS 0004 DS 0005	ADD_SUSPEND DELETE_SUSPEND SUSPEND RESUME WAIT_MVS WAIT_OLDW WAIT_OLDC	YES YES YES YES YES NO NO

DSAT gate, ATTACH function

The ATTACH function of the DSAT gate is used to attach a new task.

- The transaction manager uses the function to attach system or nonsystem tasks that have PCT entries.
- Other parts of CICS use the function to attach system tasks that do not have PCT entries.

This function is used to attach a new task, and add it to the appropriate Dispatcher queue.

When the task is first dispatched, the calling domain receives the TASK_REPLY call at its DSAT gate (see "DSAT format, TASK_REPLY function" on page 717).

Input parameters

PRIORITY
affects a task's dispatching precedence. It can have a value in the range 0 (low priority) through 255 (high priority).

USER_TOKEN
is the token by which the task to be attached is known to the caller.

[TIMEOUT]
is the deadlock time-out interval, in milliseconds.

TYPE
is the type of task. It can have either of these values:

`SYSTEM|NON_SYSTEM`

[MODE]
specifies the mode in which the task is to run. It can have any of these values:

```
CO (concurrent)
FO (file-owning)
QR (quasi-reentrant)
RO (resource-owning)
RP (ONC/RPC-owning)
SZ (secondary LU usage)
```

[TASK_REPLY_GATE_INDEX]
is used when a gate other than the attaching domain's default gate is to receive a resultant TASK_REPLY.

[SPECIAL_TYPE(SMSY)]
identifies the special task SMSY.

[TRANSACTION_TOKEN]
identifies the transaction associated with the attached task.

Output parameters

TASK_TOKEN
is the token by which the attached task is known to the dispatcher.

RESPONSE
is the dispatcher's response to the call. It can have any one of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INSUFFICIENT_STORAGE
DISASTER	LOOP ABEND USER_TASK_SLOT_UNAVAILABLE

DSAT gate, CHANGE_MODE function

The CHANGE_MODE function of DSAT gate is used to move a task from one CICS-managed TCB to another, or to select the mode in which the task is to run.

Input parameters

MODE
is the mode to be used by the task. It can have any of these values:

```
CO (concurrent)
FO (file-owning)
QR (quasi-reentrant)
RO (resource-owning)
RP (ONC/RPC-owning)
SZ (secondary LU usage)
```

[CONDITIONAL]
states whether the CHANGE_MODE should be conditional on the current load on the CPU. It can have either of these values:

`YES|NO`

MODENAME
2-character mode name.

MODENAME_TOKEN
token representing modename. More efficient than using MODENAME. The token is returned by ACTIVATE_MODE and by CHANGE_MODE (see OLD_MODENAME_TOKEN below)

TCB_TOKEN token representing the TCB instance to which to switch. The token is returned by CHANGE_MODE (see OLD_TCB_TOKEN below)

FRESH_TCB indicates whether a fresh TCB is required. It can have either of these values:

YES|NO

[PRIMARY MATCH]
an 8–byte token to be used to search for a matching free TCB instance to which to switch.

[SECONDARY MATCH]
an 8–byte token to be used to search for a matching free TCB instance to which to switch.

[MATCH_STRATEGY]
the strategy to be followed if a TCB instance that satisfies the PRIMARY_MATCH and SECONDARY_MATCH values is not found. The only value allowed is:

EXACT_THEN_NEW_THEN_BEST

Output parameters

OLD_MODE is the mode used by the task when the CHANGE_MODE request was issued. It can have any of these values:

CO (concurrent)
FO (file-owning)
QR (quasi-reentrant)
RO (resource-owning)
RP (ONC/RPC-owning)
SZ (secondary LU usage)

OLD_MODENAME is the mode used by the task when the CHANGE_MODE request was issued. It can have the same values as OLD_MODE. OLD_MODENAME is preferred to OLD_MODE.

OLD_MODENAME_TOKEN
is a token representing the mode used by the task when the CHANGE_MODE request was issued.

OLD_TCB_TOKEN
is a token representing the TCB used by the task when the CHANGE_MODE request was issued.

[MATCH_RESULT]
indicates the level of success of the matching process. It can have any of these values:

EXACT_MATCH
NO_MATCH
PRIM_NOT_SEC_MATCH
NOT_APPLIC

[NEW_TCB_TOKEN]
token representing the TCB instance returned by the matching process.

RESPONSE is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|EXCEPTION|INVALID|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, INVALID, or PURGED. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOCK_FAILED ACTIVATE_MODE_FAILED ADD_TCB_FAILED SUSPEND_FAILED
EXCEPTION	MODE_NOT ACTIVE NO_TCBS_ACTIVE INSUFFICIENT_STORAGE, TCB_FAILED TOO_FEW_TCBS

RESPONSE	Possible REASON values
INVALID	INVALID_MODENAME INVALID_MODENAME_TOKEN INVALID_TCB_TOKEN INVALID_FRESH_TCB_USAGE
PURGED	TIMED_OUT TASK_CANCELLED

DSAT gate, CLEAR_MATCH function

The CLEAR_MATCH function of the DSAT gate causes all match tokens associated with the calling TCB to be discarded.

Input parameters

None

Output parameters

RESPONSE is the dispatcher's response to the call. It can have only one value:

OK

DSAT gate, CHANGE_PRIORITY function

The CHANGE_PRIORITY function of DSAT gate has two effects:

1. It changes the dispatch priority of the issuing task.
2. It causes control to be given up to another task.

Input parameters

[PRIORITY] is the new priority. It can have a value in the range 0 (low priority) through 255 (high priority).

Output parameters

[OLD_PRIORITY]
is the task's former priority. It can have a value in the range 0 (low priority) through 255 (high priority).

RESPONSE is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|INVALID|KERNERROR

DSAT gate, SET_PRIORITY function

The SET_PRIORITY function of DSAT gate changes the priority of the issuing task, or the task specified by the TASK_TOKEN parameter.

Input parameters

PRIORITY is the new priority. It can have a value in the range 0 (low priority) through 255 (high priority).

[TASK_TOKEN] identifies the task whose priority is to be changed.

[SPECIAL_TYPE(IMMEDIATE_SHUTDOWN_TASK)]
identifies the special task "IMMEDIATE_SHUTDOWN_TASK":

Output parameters

[OLD_PRIORITY]
is the task's former priority. It can have a value in the range 0 (low priority) through 255 (high priority).

[RESPONSE] is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR

[REASON] is returned when RESPONSE is EXCEPTION. It has the value:

INVALID_TASK_TOKEN

DSAT gate, CANCEL_TASK function

The CANCEL_TASK function of DSAT gate causes a specified task to be canceled. The task is cancelled when in a suitable suspend or when a deferred abend can be delivered to the task.

Input parameters

TASK_TOKEN
is the token representing the task to be canceled.

CANCEL_TYPE
can have either of these values:

FORCE_CANCEL|NORMAL_CANCEL

DEFERRED_ABEND_CODE
is the abend code to be used when the task is abended during deferred abend processing.

Output parameters

RESPONSE
is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_TASK_TOKEN INVALID_STATE NOT_PURGEABLE, CANCEL_INHIBITED INVALID_STATE_PURGE

DSAT gate, FREE_SUBSPACE_TCBS function

The FREE_SUBSPACE_TCBS function of DSAT gate releases any open subspace TCBs owned by the task, and makes them available for use by another task executing with the same subspace, or deletes the TCBs if the task is 'unclean'.

Input parameters

None

Output parameters

OPEN_TCBS_USED_AND_KEPT
is a bit string indicating which TCB modes were used by the task, of and are now available to other tasks

OPEN_TCBS_USED_AND_LOST
is a bit string indicating which TCB modes were used by the task, of and have now been deleted because the task was 'unclean'

RESPONSE
is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOCK_FAILED
INVALID	NOT_SUBSPACE_ELIGIBLE

DSAT gate, DELETE_SUBSPACE_TCBS function

The DELETE_SUBSPACE_TCBS function of DSAT gate deletes any open TCBs associated with the given subspace.

Input parameters

SUBSPACE_TOKEN
indicates the subspace whose associated open TCBs are to be deleted

Output parameters

RESPONSE is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|EXCEPTION|KERNERROR

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOCK_FAILED
EXCEPTION	TOO_FEW_TCBS

DSAT gate, TCB_POOL_MANAGEMENT function

The TCB_POOL_MANAGEMENT function of DSAT gate deletes unallocated TCBs which are excess to current requirements.

Input parameters

None

Output parameters

RESPONSE is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|KERNERROR

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOCK_FAILED

DSAT gate, RELEASE_OPEN_TCB function

The RELEASE_OPEN_TCB function of DSAT gate frees the TCB from the calling task's ownership.

Input parameters

TCB_TOKEN is the token representing the task that owns the TCB to be freed.

Output parameters

RESPONSE is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOCK_FAILED
INVALID	INVALID_TCB_TOKEN, TCB_NOT_OWNED

DSAT gate, SET_TRANSACTION_TOKEN function

The SET_TRANSACTION_TOKEN function of DSAT gate sets the XM domain transaction token of the transaction associated with the currently dispatched task.

Input parameters

TRANSACTION_TOKEN
identifies the dispatcher task for which SET_TRANSACTION_TOKEN is to be performed.

Output parameters

RESPONSE is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|INVALID|KERNERROR

DSBR gate, START_BROWSE function

The START_BROWSE function of DSBR gate starts a browse session with the dispatcher.

Input parameters

None.

Output parameters

BROWSE_TOKEN is the token representing this browse session.

RESPONSE is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|PURGED

DSBR gate, END_BROWSE function

The END_BROWSE function of DSBR gate ends a browse session with the dispatcher.

Input parameters

BROWSE_TOKEN is the token identifying the browse session to be ended.

Output parameters

RESPONSE is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON] is returned when RESPONSE is INVALID. It has this value:

INVALID_BROWSE_TOKEN

DSBR gate, GET_NEXT function

The GET_NEXT function of DSBR gate returns information about the next task.

Input parameters

BROWSE_TOKEN is the token identifying the browse session.

Output parameters

[TASK_TOKEN] is the token by which the task is known to the dispatcher.

[KERNEL_TOKEN]
is the token by which the task is known to the kernel.

[DOMAIN_INDEX]
is the 2-character index identifying the domain that made the ATTACH call for the task.

[OPEN_MODES] is a 32-bit string which indicates which modes of open TCBs were used by this task.

[PRIORITY] is the task's dispatch priority. It can have a value in the range 0 (low priority) through 255 (high priority).

[TYPE] is the type of task. It can have either of these values:

SYSTEM|NON_SYSTEM

[STATE] is the state of the task. It can have any one of these values:

READY|RUNNING|SUSPENDED

[RESOURCE_NAME]
is the name of the resource that the task is waiting for, if the task is suspended.

[RESOURCE_TYPE]
is the type of resource that the task is waiting for, if the task is suspended.

[RESOURCE_TIME]
is the interval of time that has passed since the task last issued a suspend or wait.

[USER_TOKEN] is the token by which the task is known to the caller that made the ATTACH request for the task.

[SUSPEND_TOKEN]
is the token by which the dispatcher recognizes a task to be suspended or resumed.

[MODE]
is the mode in which the task is to run. It can have any one of these values:

```
CO (concurrent)
FO (file-owning)
QR (quasi-reentrant)
RO (resource-owning)
RP (ONC/RPC-owning)
SZ (secondary LU usage)
```

[TCB_TYPE]
is the type of TCB that the task is executing on. It can have any one of these values:

`CKOPEN_TCB|UKOPEN_TCB|QR_TCB|INTERNAL_TCB`

[TCB_TOKEN]
is the TCB token associated with the task.

[ESSENTIAL_TCB]
indicates whether the TCB is an essential TCB or not. It can have either of these values:

`ESSENTIAL_YES|ESSENTIAL_NO`

RESPONSE
is the dispatcher's response to the call. It can have any one of these values:

`OK|DISASTER|EXCEPTION|INVALID|KERNERROR`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	END, INVALID_BROWSE_TOKEN

DSBR gate, INQUIRE_TASK function

The INQUIRE_TASK function of DSBR gate returns information about a specified task.

Input parameters

[INPUT_TASK_TOKEN]
is the token for the task to be inquired on.

Output parameters

[TASK_TOKEN]
is the token by which the task is known to the dispatcher.

[KERNEL_TOKEN]
is the token by which the task is known to the kernel.

[DOMAIN_INDEX]
is the 2-character index identifying the domain that made the ATTACH call for the task.

[OPEN_MODES]
is a 32-bit string which indicates which modes of open TCBs were used by this task.

[PRIORITY]
is the task's dispatch priority. It can have a value in the range 0 (low priority) through 255 (high priority).

[TYPE]
is the type of task. It can have either of these values:

`SYSTEM|NON_SYSTEM`

[STATE]
is the state of the task. It can have any one of these values:

`READY|RUNNING|SUSPENDED`

[RESOURCE_NAME]
is the name of the resource that the task is waiting for, if the task is suspended.

[RESOURCE_TYPE]
is the type of resource that the task is waiting for, if the task is suspended.

[RESOURCE_TIME]
is the interval of time that has passed since the task last issued a suspend or wait.

[USER_TOKEN]
is the token by which the task is known to the caller that made the ATTACH request for the task.

[SUSPEND_TOKEN]
is the token by which the dispatcher recognizes a task to be suspended or resumed.

[MODE]
is the mode in which the task is to run. It can have any one of these values:

CO (concurrent)
FO (file-owning)
QR (quasi-reentrant)
RO (resource-owning)
RP (ONC/RPC-owning)
SZ (secondary LU usage)

[TCB_TYPE] is the type of TCB that the task is executing on. It can have any one of these values:

CKOPEN_TCB|UKOPEN_TCB|QR_TCB|INTERNAL_TCB

[TCB_TOKEN] is the TCB token associated with the task.

[ESSENTIAL_TCB]
indicates whether the TCB is an essential TCB or not. It can have either of these values:

ESSENTIAL_YES|ESSENTIAL_NO

[CANCEL_PENDING]
Not supported by domain gate function.

[DEFERRED_ABEND_CODE]
Not supported by domain gate function.

RESPONSE is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_SUPPORTED
INVALID	INVALID_TASK_TOKEN

DSBR gate, SET_TASK function

The SET_TASK function of DSBR gate marks the task as "unclean" so that open TCBs will be freed at task termination.

Input parameters

[INPUT_TASK_TOKEN]
is the token by which the task is known to the dispatcher.

[CLEANLINESS]
specifies that the task is to be marked "unclean". It can take only the value UNCLEAN.

[ABTERM_ALLOWED]
Not supported by domain gate function.

[WAIT] Not supported by domain gate function.

[CLEAR_CANCEL_PENDING]
Not supported by domain gate function.

Output parameters

[ACTION] Not supported by domain gate function.

RESPONSE is the dispatcher's response to the call. It can have any one of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

INVALID_TASK_TOKEN, NOT_SUPPORTED

DSBR gate, INQUIRE_TCB function

The INQUIRE_TCB function of the DSBR gate returns the AP TCB-related token associated with the specified DS TCB_TOKEN. If the AP token has not yet been set by SET_TCB, then the function returns an AP_TCB_TOKEN value of zero.

Input parameters

[TCB_TOKEN] token provided by DS representing the TCB instance for which the associated AP-related token is required. If this is omitted, the token of the running TCB is assumed.

Output parameters

OWNER_TCB_TOKEN
token, provided by the TCB's owning domain, associated with the TCB instance defined by TCB_TOKEN.

RESPONSE
is the dispatcher's response to the call. It can have one of these values:

OK|INVALID

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_TCB_TOKEN

DSBR gate, SET_TCB function

The SET_TCB function of the DSBR gate sets the AP TCB-related token to be associated with the running TCB.

Input parameters

OWNER_TCB_TOKEN
token, provided by the TCB's owning domain, to be associated with the running TCB.

Output parameters

RESPONSE
is the dispatcher's response to the call. It can have only one value:

OK

DSIT gate, INQUIRE_DISPATCHER function

The INQUIRE_DISPATCHER function of DSIT gate returns information about the current state of the dispatcher.

Input parameters

None.

Output parameters

[NUMBER_OF_SUBTASKS]
is the number of subtasks for concurrent mode.

[SCAN_DELAY_INTERVAL]
is the delay before terminal control is dispatched after a terminal is posted by the access method.

[MAXIMUM_WAIT_INTERVAL]
is the maximum delay before terminal control is dispatched.

[PRIORITY_MULTIPLIER]
determines how the priority of new tasks is to be penalized in 'storage getting short' and 'storage critical' situations.

[QR_BATCHING_VALUE]
is the number of POSTs for BATCH=YES waits in quasi-reentrant mode.

[MAXOPENTCBS]
is the maximum number of TCBs in the TCB pool known as the **open pool**.

[ACTOPENTCBS]
is the number of TCBs in the TCB pool known as the **open pool** which are being used by current tasks.

[RP_TCB_ATTACHED]
indicates whether or not the RP TCB is attached. It can have either of these values:

YES|NO

[SZ_TCB_ATTACHED]
indicates whether or not the SZ TCB is attached. It can have either of these values:

YES|NO

MAXJVMTCBS is the maximum number of TCBs in the JVM TCB pool.
MAXHPTCBS is the maximum number of TCBS in the hotpooling Java TCB pool.
ACTJVMTCBS is the number of TCBs in the JVM TCB pool which are being used by current tasks.
ACTHPTCBS is the number of TCBS in the hotpooling Java TCB pool which are being used by current tasks.
RESPONSE is the dispatcher's response to the call. It can have either of these values:
OK|DISASTER

DSIT gate, SET_DISPATCHER function

The SET_DISPATCHER function of DSIT gate sets the state of the dispatcher.

Input parameters

[NUMBER_OF_SUBTASKS]
is the number of subtasks for concurrent mode.
[SCAN_DELAY_INTERVAL]
is the delay before terminal control is dispatched after a terminal is posted by the access method.
[MAXIMUM_WAIT_INTERVAL]
is the maximum delay before terminal control is dispatched.
[PRIORITY_MULTIPLIER]
determines how quickly a task's priority increases as it waits to be dispatched. The faster it increases the less likely a low priority task is to be held up for long periods by higher priority tasks in a busy system.
[QR_BATCHING_VALUE]
is the number of POSTs for BATCH=YES waits in quasi reentrant mode.
[MAXOPENTCBS]
is the maximum number of TCBs in the TCB pool known as the **open pool**.
MAXJVMTCBS is the maximum number of TCBs in the JVM TCB pool.
MAXHPTCBS is the maximum number of TCBS in the hotpooling Java TCB pool.

Output parameters

RESPONSE is the dispatcher's response to the call. It can be any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR
[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	MAXWAIT_LESSTHAN_SCANDELAY MAXOPENTCBS_OUT_OF_RANGE TOO_LATE_TO_SET_SUBTASKS MAXJVMTCBS_OUT_OF_RANGE MAXHPTCBS_OUT_OF_RANGE

DSIT gate, ACTIVATE_MODE function

The ACTIVATE_MODE function creates a mode to which TCBs can be added (by ADD_TCB) so that tasks can CHANGE_MODE to the TCBs.

Input parameters

MODE is the mode to be activated. MODE and MODENAME are mutually exclusive but one must be specified. Mode is a single byte whose values correspond to the pre 1.3 modes QR, RO, FO, CO, SZ and RP.
MODENAME is the two character string that becomes the block name of the sub dispatcher block created by ACTIVATE_MODE.
IDENTITY is the name of the mode to be activated. It is a two byte character string.
EXEC_CAPABLE indicates whether TCBs in this mode are to be set up to support the use of EXEC CICS commands by code running on them.

LE_ENVIRONMENT
indicates whether Language Environment is to run in native MVS mode or in CICS mode on TCBs in this mode.

TCB_KEY
indicates the key to be specified on ATTACHes of TCBs in this mode.

INHERIT_SUBSPACE
indicates whether TCBs in this mode will be able to run application code in a subspace.

ESSENTIAL_TCB
indicates whether CICS is to be brought down if a TCB in this mode suffers a non recoverable abend.

PRTY_RELATIVE_TO_QR
allows TCBs in this mode to have a different priority to that of the QR TCB.

MULTIPLE_TCBS
indicates whether this mode allows more than one TCB.

OPEN
indicates whether TCBs in this mode are to be managed by the Dispatcher domain as "Open TCBs".

[NOTIFY_DELETE]
indicates which domain, if any, to notify when a DELETE_TCB is issued. It is the binary domain index for the domain.

[OPEN_POOL_NUMBER]
is the number of the open TCB pool which is to contain TCBs of the newly-activated mode.

[DEPENDENT_ON]
indicates that TCBs of the mode being activated depend on the existence of TCBs of another mode.

[SZERO]
indicates whether TCBs of the new mode should be attached with SZERO(YES) or SZERO(NO). It can have either of these values:

SZERO_YES|SZERO_NO

[WAIT_FOR_MATCH]
indicates if a CHANGE_MODE should consider waiting for a suitable TCB rather than using a free TCB. It can have one of these values:

NO_PRIMARY|NO_MODE|NEVER

Output parameters

[MODENAME_TOKEN]
is a token that identifies this modename.

RESPONSE
is the dispatcher's response to the call. It can be any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	MODE_ALREADY_ACTIVE INSUFFICIENT_STORAGE, MODENAME_ALREADY_ACTIVE RESERVED_MODENAME MODE_LIMIT_REACHED TOO_MANY_MULTI
INVALID	INVALID_MODE INVALID_POOL_NUMBER

DSIT gate, ADD_TCB function

The ADD_TCB function adds a TCB to a particular mode.

Input parameters

MODENAME
specifies the name of the mode the TCB is to be added to. MODENAME and MODENAME_TOKEN are mutually exclusive but one of them must be coded.

MODENAME_TOKEN
identifies mode the TCB is to be added by using the token returned by the ACTIVATE_MODE.

IDENTITY
is an eight character string to placed in the Dispatcher's block that represents the TCB.

Output parameters

TCB_TOKEN
is a token that uniquely idenifies this TCB.

RESPONSE
is the dispatcher's response to the call. It can be any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INSUFFICIENT_STORAGE RESERVED_MODENAME MODE_LIMIT_REACHED MODE_NOT_ACTIVE
INVALID	INVALID_MODENAME INVALID_MODENAME_TOKEN

DSIT gate, DELETE_TCB function

The DELETE_TCB function is used by the caller to tell the Dispatcher that the TCB is to be shutdown and that the associated control blocks can be freed. If an attempt is made to shut down an essential TCB, an EXCEPTION response is returned with a reason of NOT_SUPPORTED.

Note that no quiescing of tasks on the TCB is performed.

Input parameters

TCB_TOKEN
is a token that uniquely identifies the TCB.

Output parameters

RESPONSE
is the dispatcher's response to the call. It can be any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_SUPPORTED TCB_IN_USE
INVALID	INVALID_TCB_TOKEN

DSIT gate, DELETE_ OPEN_TCB function

DELETE_OPEN_TCB schedules the termination of an open TCB. If the TCB is currently in use, the termination will occur when the owning task terminates.

Input parameters

TCB_TOKEN
is a token provided by DS that uniquely identifies the TCB.

Output parameters

RESPONSE
is the dispatcher's response to the call. It can be any one of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_TCB_TOKEN

DSIT gate, DELETE_ALL_OPEN_TCBS function

DELETE_ALL_OPEN_TCBS schedules the termination of all open TCBs with a given modename. For TCBs that are currently in use, the termination will occur when the owning task terminates. The function does not prevent new TCBs of the given mode from being created.

Input parameters

MODENAME is the name of the mode of the TCBs to be deleted.

Output parameters

RESPONSE is the dispatcher's response to the call. It can be any one of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	MODE_NOT_ACTIVE
INVALID	INVALID_MODENAME

DSIT gate, FREE_TCB function

The FREE_TCB function is issued by the Kernel and tells the Dispatcher that a given TCB has terminated and been DETACHed.

Input parameters

TCB_TOKEN is a token that uniquely identifies the TCB.

Output parameters

RESPONSE is the dispatcher's response to the call. It can be any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TASK_NOT_TERMINATED INVALID_TCB_TOKEN

DSIT gate, PROCESS_DEAD_TCBS function

The PROCESS_DEAD_TCBS function is issued by the SM system task each time it runs to tell the Dispatcher to process any TCBs it finds on its dead TCB chain. Such TCBs will be in an MVS WAIT issued by their ESTAE exit after suffering a non recoverable abend.

Input parameters

None.

Output parameters

RESPONSE is the dispatcher's response to the call. It can be any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

DSSR gate, ADD_SUSPEND function

The ADD_SUSPEND function of DSSR gate returns a suspend token which is used to identify a task to be suspended or resumed.

Input parameters

[RESOURCE_NAME]
is the name of the resource that the task is suspended on.

[RESOURCE_TYPE]
is the type of resource that the task is suspended on.

Output parameters

SUSPEND_TOKEN
is the token that is used to identify the task to be suspended or resumed.

RESPONSE
is the dispatcher's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. It has this value:

INSUFFICIENT_STORAGE

DSSR gate, DELETE_SUSPEND function

The DELETE_SUSPEND function of DSSR gate discards a suspend token.

Input parameters

SUSPEND_TOKEN
is the suspend token to be deleted.

Output parameters

RESPONSE
is the dispatcher's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_SUSPEND_TOKEN SUSPEND_TOKEN_IN_USE

DSSR gate, SUSPEND function

The SUSPEND function of DSSR gate causes a running task to be suspended.

Input parameters

Note: [INTERVAL] and [DEADLOCK_ACTION] are ***mutually exclusive*** parameters.

SUSPEND_TOKEN
is the token identifying the task to be suspended.

PURGEABLE
is the purgeable status of the task. It can have either of these values:

YES|NO

[INTERVAL]
is an interval (in units as specified by TIME_UNIT) after which the task is given back control if it has not been resumed by a DSSR RESUME call.

[DEADLOCK_ACTION]
describes whether the suspended task should be purged if deadlock is detected, and if so, how it should be purged. It can have any one of these values:

DELAYED|IMMEDIATE|INHIBIT

[RESOURCE_NAME]
is the name of the resource that the task is suspended on.

[RESOURCE_TYPE]
is the type of resource that the task is suspended on.

[TIME_UNIT]
identifies the time units specified on the INTERVAL and DELAY parameters where present. It can have either of these values:

SECOND|MILLI_SECOND

[DELAY]
is an interval (in units as specified by TIME_UNIT) during which the task is not dispatched if CICS has other work to do.

[RETRY]
indicates whether or not the dispatcher is to retry the suspend operation, if the running task is not suspended by a preceding suspend operation. It can have either of these values:

YES|NO

[WLM_WAIT_TYPE]
indicates the reason for task's wait state to the MVS workload manager. It can have any of these values:

LOCK|IO|CONV|CMDRESP|DISTRIB|
SESS_LOCALMVS|SESS_NETWORK|
SESS_SYSPLEX|TIMER|OTHER_PRODUCT|
MISC|IDLE

[DISPATCH_BEFORE_WAIT]
indicates if the suspended task is prepared to wait across a partition exit It can have either of these values:

YES|NO

[TEMP_HIGH_PRIORITY]
indicates if the task is to get a temporary priority boost at the completion of the suspend. It can have either of these values:

YES|NO

Output parameters

[COMPLETION_CODE]
is a completion code supplied by the resumed task.

RESPONSE
is the dispatcher's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or PURGED. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_SUSPEND_TOKEN ALREADY_SUSPENDED CLEAN_UP_PENDING
PURGED	TASK_CANCELLED TIMED_OUT

DSSR gate, RESUME function

The RESUME function of DSSR gate causes a suspended task to be resumed.

Input parameters

SUSPEND_TOKEN
is the token identifying the task to be resumed.

[COMPLETION_CODE]
is a completion code to be passed from the resumed task to the suspended task.

Output parameters

RESPONSE
is the dispatcher's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TASK_CANCELLED TIMED_OUT

RESPONSE	Possible REASON values
INVALID	INVALID_SUSPEND_TOKEN ALREADY_RESUMED

DSSR gate, WAIT_MVS function

The WAIT_MVS function of DSSR gate causes a task to wait on an ECB, or list of ECBs, to be posted via the MVS POST service.

Input parameters

Note: ECB_ADDRESS and ECB_LIST_ADDRESS are ***mutually exclusive*** parameters; [INTERVAL] and [DEADLOCK_ACTION] are also ***mutually exclusive***.

ECB_ADDRESS
is the address of the ECB for the task.

ECB_LIST_ADDRESS
is the address of a list of ECBs for the task.

PURGEABLE
is the purgeable status of the task. It can have either of these values:

YES|NO

[INTERVAL]
is an interval (in units as specified by TIME_UNIT) after which the task is given back control if it has not been resumed by a DSSR RESUME call.

[DEADLOCK_ACTION]
describes whether the suspended task should be purged if deadlock is detected, and if so, how it should be purged. It can have any one of these values:

DELAYED|IMMEDIATE|INHIBIT

[RESOURCE_NAME]
is the name of the resource that the task is suspended on.

[RESOURCE_TYPE]
is the type of resource that the task is suspended on.

[BATCH]
states whether requests are to be batched. It can have either of these values:

YES|NO

[TIME_UNIT]
identifies the time units specified on the INTERVAL and DELAY parameters where present. It can have either of these values:

SECOND|MILLI_SECOND

[DELAY]
is an interval (in units as specified by TIME_UNIT) during which the task is not dispatched if CICS has other work to do.

[RETRY]
indicates whether or not the dispatcher is to retry the suspend operation, if the running task is not suspended by a preceding suspend operation. It can have either of these values:

YES|NO

[WLM_WAIT_TYPE]
indicates the reasonfor task's wait state to the MVS workload manager. It can have any of these values:

LOCK|IO|CONV|CMDRESP|DISTRIB|
SESS_LOCALMVS|SESS_NETWORK|
SESS_SYSPLEX|TIMER|OTHER_PRODUCT|
MISC|IDLE

[DISPATCH_BEFORE_WAIT]
indicates if the suspended task is prepared to wait across a partition exit It can have either of these values:

YES|NO

[TEMP_HIGH_PRIORITY]
indicates if the task is to get a temporary priority boost at the completion of the suspend. It can have either of these values:

YES|NO

Output parameters

RESPONSE
is the dispatcher's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or PURGED. Possible values are:

RESPONSE	Possible REASON values
INVALID	ALREADY_WAITING INVALID_ECB_ADDR
PURGED	TASK_CANCELLED TIMED_OUT

DSSR gate, WAIT_OLDW function

The WAIT_OLDW function of DSSR gate causes a task to wait on an ECB, or list of ECBs, that may be posted via the MVS POST service or by setting the POST bit (X'40' in the first byte). This is supported only in QR mode.

Input parameters

Note: ECB_ADDRESS and ECB_LIST_ADDRESS are ***mutually exclusive*** parameters; [INTERVAL] and [DEADLOCK_ACTION] are also ***mutually exclusive***.

ECB_ADDRESS
is the address of the ECB for the task.

ECB_LIST_ADDRESS
is the address of a list of ECBs for the task.

PURGEABLE
is the purgeable status of the task. It can have either of these values:

YES|NO

[INTERVAL]
is an interval (in units as specified by TIME_UNIT) after which the task is given back control if it has not been resumed by a DSSR RESUME call.

[DEADLOCK_ACTION]
describes whether the suspended task should be purged if deadlock is detected, and if so, how it should be purged. It can have any one of these values:

DELAYED|IMMEDIATE|INHIBIT

[RESOURCE_NAME]
is the name of the resource that the task is suspended on.

[RESOURCE_TYPE]
is the type of resource that the task is suspended on.

[SPECIAL_TYPE(CSTP)]
identifies the special task CSTP.

[TIME_UNIT]
identifies the time units specified on the INTERVAL and DELAY parameters where present. It can have either of these values:

SECOND|MILLI_SECOND

[DELAY]
is an interval (in units as specified by TIME_UNIT) during which the task is not dispatched if CICS has other work to do.

[RETRY]
indicates whether or not the dispatcher is to retry the suspend operation, if the running task is not suspended by a preceding suspend operation. It can have either of these values:

YES|NO

[WLM_WAIT_TYPE]
indicates the reason for task's wait state to the MVS workload manager. It can have any of these values:

LOCK|IO|CONV|CMDRESP|DISTRIB|
SESS_LOCALMVS|SESS_NETWORK|
SESS_SYSPLEX|TIMER|OTHER_PRODUCT|
MISC|IDLE

[DISPATCH_BEFORE_WAIT]
indicates if the suspended task is prepared to wait across a partition exit It can have either of these values:

YES|NO

[TEMP_HIGH_PRIORITY]
indicates if the task is to get a temporary priority boost at the completion of the suspend. It can have either of these values:

YES|NO

Output parameters

RESPONSE
is the dispatcher's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or PURGED. Possible values are:

RESPONSE	Possible REASON values
PURGED	TASK_CANCELLED TIMED_OUT
INVALID	ALREADY_WAITING, INVALID_ECB_ADDR, INVALID_MODE

DSSR gate, WAIT_OLDC function

The WAIT_OLDC function of DSSR gate causes a task to wait on an ECB that must be posted by setting the X'40' bit rather than via the MVS POST service. This is supported only in QR mode.

Input parameters

Note: [INTERVAL] and [DEADLOCK_ACTION] are ***mutually exclusive*** parameters.

ECB_ADDRESS
is the address of the ECB for the task.

PURGEABLE
is the purgeable status of the task. It can have either of these values:

YES|NO

[INTERVAL]
is an interval (in units as specified by TIME_UNIT) after which the task is given back control if it has not been resumed by a DSSR RESUME call.

[DEADLOCK_ACTION]
describes whether the suspended task should be purged if deadlock is detected, and if so, how it should be purged. It can have any one of these values:

DELAYED|IMMEDIATE|INHIBIT

[RESOURCE_NAME]
is the name of the resource that the task is suspended on.

[RESOURCE_TYPE]
is the type of resource that the task is suspended on.

[TIME_UNIT]
identifies the time units specified on the INTERVAL and DELAY parameters where present. It can have either of these values:

SECOND|MILLI_SECOND

[DELAY]
is an interval (in units as specified by TIME_UNIT) during which the task is not dispatched if CICS has other work to do.

[RETRY]
indicates whether or not the dispatcher is to retry the suspend operation, if the running task is not suspended by a preceding suspend operation. It can have either of these values:

YES|NO

[WLM_WAIT_TYPE]
indicates the reason for task's wait state to the MVS workload manager. It can have any of these values:

LOCK|IO|CONV|CMDRESP|DISTRIB|
SESS_LOCALMVS|SESS_NETWORK|
SESS_SYSPLEX|TIMER|OTHER_PRODUCT|
MISC|IDLE

[DISPATCH_BEFORE_WAIT]
indicates if the suspended task is prepared to wait across a partition exit It can have either of these values:

YES|NO

[TEMP_HIGH_PRIORITY]
indicates if the task is to get a temporary priority boost at the completion of the suspend. It can have either of these values:

YES|NO

Output parameters

RESPONSE
is the dispatcher's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or PURGED. Possible values are:

RESPONSE	Possible REASON values
INVALID	ALREADY_WAITING INVALID_ECB_ADDR INVALID_MODE
PURGED	TASK_CANCELLED TIMED_OUT

Dispatcher domain's generic gates

Table 47 summarizes the dispatcher domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 47. Dispatcher domain's generic gates

Gate	Trace	Function	Format
DMDM	DS 0006 DS 0007	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
KEDS	DS 0012 DS 0013	TCB_REPLY TASK_REPLY	KEDS
SMNT	DS 0145 DS 0113	STORAGE_NOTIFY	SMNT
STST	DS 0020 DS 0021	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
APUE	DS 0121 DS 0122	SET_EXIT_STATUS	APUE

For descriptions of these functions and their input and output parameters, you need to refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format APUE—"Application domain's generic formats" on page 593
Format DMDM—"Domain manager domain's generic formats" on page 669
Format KEDS—"Kernel domain's generic formats" on page 848
Format STST—"Statistics domain's generic format" on page 1198
Format SMNT—"Storage manager domain's generic formats" on page 1159

In preinitialization processing, the dispatcher domain sets the initial dispatching options:
- The priority aging value (PRTYAGE)
- Whether or not tasks are to be run in concurrent mode (SUBTSKS)
- The terminal scan delay interval (ICVTSD)
- The region exit time (ICV).

For a cold start, the information comes from the system initialization parameters (given in parentheses); for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

Dispatcher domain's generic formats

Table 48 describes the generic formats owned by the dispatcher domain and shows the functions performed on the calls.

Table 48. Generic formats owned by dispatcher domain

Format	Calling modules	Functions
DSAT	DFHDSKE DFHDSDS4 DFHSJIN DFHSMVN	TASK_REPLY PURGE_INHIBIT_QUERY FORCE_PURGE_INHIBIT_QUERY NOTIFY_DELETE_TCB

In the descriptions of the formats that follow, the "input" parameters are input not to the dispatcher, but to the domain being called by the dispatcher. Similarly, the "output" parameters are output by the domain that was called by the dispatcher, in response to the call.

DSAT format, TASK_REPLY function

The TASK_REPLY function of DSAT format is used to notify the domain that attached a task that the task has had its first dispatch.

Input parameters

USER_TOKEN is the token by which the task that has been dispatched is known to the called domain.

TASK_TOKEN is the token by which the task that has been dispatched is known to the dispatcher.

SUSPEND_TOKEN
is the suspend token that the task can be suspended against by default.

Output parameters

RESPONSE is the called domain's response to the call. It can have any one of these values:

OK|DISASTER|INVALID|KERNERROR

DSAT format, PURGE_INHIBIT_QUERY function

The PURGE_INHIBIT_QUERY function of DSAT format is used by the dispatcher to see if a task selected for purge can be purged. Its main purpose is to find out from the AP domain whether the task is currently purgeable by the system.

Input parameters

USER_TOKEN
is the token by which the task that has been dispatched is known to the called domain.

TASK_TOKEN
is the token by which the task that has been dispatched is known to the dispatcher.

Output parameters

PURGE_INHIBITED_RESPONSE
states whether the task can be purged. It can have either of these values:

YES|NO

RESPONSE
always has the value OK.

DSAT format, FORCE_PURGE_INHIBIT_QUERY function

The FORCE_PURGE_INHIBIT_QUERY function of DSAT format is used by the dispatcher to see if a task selected for purge can be forcepurged. Its main purpose is to find out from the AP domain whether the task is currently purgeable by the system.

Input parameters

USER_TOKEN
is the token by which the task that has been dispatched is known to the called domain.

TASK_TOKEN
is the token by which the task that has been dispatched is known to the dispatcher.

Output parameters

PURGE_INHIBITED_RESPONSE
states whether the task can be purged. It can have either of these values:

YES|NO

RESPONSE
always has the value OK.

DSAT format, NOTIFY_DELETE_TCB function

The NOTIFY_DELETE function of DSAT format notifies the interested domain (as specified in the NOTIFY_DELETE parameter on the DSIT ACTIVATE_MODE request for the mode) that a DELETE_TCB request is in progress.

Input parameters

TCB_TOKEN
The DS token representing the TCB instance for which notification is required when deleted.

Output parameters

RESPONSE
is the dispatcher's response to the call. It can have only one value:

OK

Modules

Module	Function
DFHDSAT	Receives calls to the dispatcher DSAT gate. This gate carries out such work as: ATTACH—Create new task CHANGE_MODE—Change mode of running task CHANGE_PRIORITY—Change priority of running task and release control SET_PRIORITY—Change priority of running task or other task and keep running CANCEL_TASK—Cancel specified task.

Module	Function
DFHDSBR	Handles the following requests: START_BROWSE GET_NEXT END_BROWSE INQUIRE_TASK
DFHDSDM	Handles the following dispatcher requests: DMDM PRE_INITIALISE DMDM INITIALISE_DOMAIN DMDM QUIESCE_DOMAIN DMDM TERMINATE_DOMAIN
DFHDSIT	Handles the following dispatcher requests: INQUIRE_DISPATCHER SET_DISPATCHER
DFHDSKE	Handles kernel DS requirements, and handles the following requests: KEDS TCB_REPLY KEDS TASK_REPLY
DFHDSSM	Receives the STORAGE_NOTIFY call from the storage manager domain.
DFHDSSR	Handles the following requests: ADD_SUSPEND DELETE_SUSPEND INQUIRE_SUSPEND_TOKEN SUSPEND RESUME WAIT_MVS WAIT_OLDW WAIT_OLDC
DFHDSST	Receives statistics calls from the ST domain
DFHDSUE	Receives the user exit gate call from the AP domain

Exits

There are two global user exit points in the dispatcher domain: XDSAWT and XDSBWT. For further information about these, see the *CICS Customization Guide*.

Trace

The point IDs for the dispatcher domain are of the form DS xxxx; the corresponding trace levels are DS 1, DS 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 81. Dump domain (DU)

The dump domain is responsible for producing storage dumps and for handling the associated data sets and status in the CICS system. Two types of dump are produced:

Transaction dumps
: These are written to the CICS-managed BSAM data sets DFHDMPA and DFHDMPB. They consist of the storage areas related to a particular transaction.

System dumps
: CICS uses the MVS SDUMP facility to dump the entire CICS region to an MVS SYS1.DUMP data set.

The two dump tables (one for each dump type) are indexed by the dump code and contain details of the options required for each request.

Design overview

Figure 108 gives an overview of the dump domain architecture.

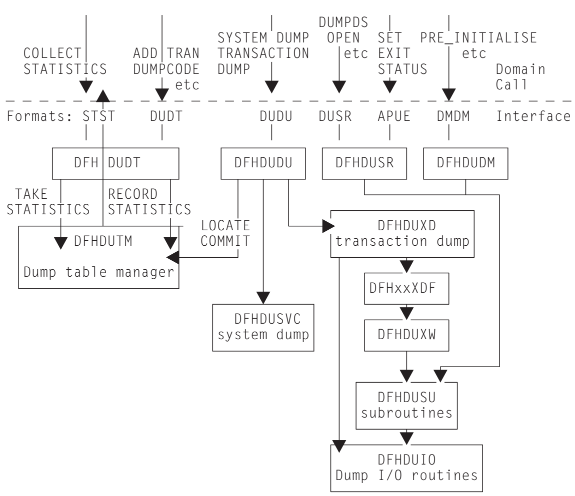


Figure 108. CICS dump domain structure

Dump domain's specific gates

Table 49 summarizes the dump domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 49. Dump domain's specific gates

Gate	Trace	Function	XPI
DUDT	DU 0500	ADD_TRAN_DUMPCODE	NO
	DU 0501	DELETE_TRAN_DUMPCODE	NO
		INQUIRE_TRAN_DUMPCODE	NO
		SET_TRAN_DUMPCODE	NO
		STARTBR_TRAN_DUMPCODE	NO
		GETNEXT_TRAN_DUMPCODE	NO
		ENDBR_TRAN_DUMPCODE	NO
		ADD_SYSTEM_DUMPCODE	NO
		DELETE_SYSTEM_DUMPCODE	NO
		INQUIRE_SYSTEM_DUMPCODE	NO
		SET_SYSTEM_DUMPCODE	NO
		STARTBR_SYSTEM_DUMPCODE	NO
		GETNEXT_SYSTEM_DUMPCODE	NO
		ENDBR_SYSTEM_DUMPCODE	NO

Table 49. Dump domain's specific gates (continued)

Gate	Trace	Function	XPI
DUDU	DU 0101	TRANSACTION_DUMP	YES
	DU 0102	SYSTEM_DUMP	YES
DUSR	DU 0301	DUMPDS_OPEN	NO
	DU 0302	DUMPDS_CLOSE	NO
		DUMPDS_SWITCH	NO
		INQUIRE_CURRENT_DUMPDS	NO
		INQUIRE_DUMPDS_OPEN_STATUS	NO
		INQUIRE_DUMPDS_AUTOSWITCH	NO
		SET_DUMPDS_AUTOSWITCH	NO
		SET_DUMPTABLE_DEFAULTS	NO
		INQUIRE_INITIAL_DUMPDS	NO
		SET_INITIAL_DUMPDS	NO
		INQUIRE_SYSTEM_DUMP	NO
		SET_SYSTEM_DUMP	NO
		INQUIRE_RETRY_TIME	NO
		SET_RETRY_TIME	NO
		SET_CONNECT_TOKEN	

DUDT gate, ADD_TRAN_DUMPCODE function

The ADD_TRAN_DUMPCODE function of the DUDT gate is invoked to add a new dump code to the transaction dump table.

Input parameters

DUMPSCOPE
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:

LOCAL|RELATED

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

TRANSACTION_DUMPCODE
is the transaction dump code.

TRANSACTION_DUMP
states whether a transaction dump is required for this dump code. It can have either of these values:

YES|NO

SYSTEM_DUMP
states whether a system dump is required for this dump code. It can have either of these values:

YES|NO

TERMINATE_CICS
states whether CICS is to be terminated for this dump code. It can have either of these values:

YES|NO

MAXIMUM_DUMPS
is the maximum number of times the dump code action can be taken.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE_DUMPCODE INVALID_DUMPCODE CATALOG_FULL INSUFFICIENT_STORAGE IO_ERROR

Process flow

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT ADD_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

DFHDUTM process flow

1. Validate the dump code to be added. Transaction dump codes are 4 bytes and must not contain leading or embedded blanks. If the dump code is not valid, return to the caller indicating the exception.
2. Scan the transaction dump table to find the correct place to insert the dump code in collating sequence. If an entry already exists for that dump code, return to the caller indicating duplicate dump code. If the entry is about to use the last available entry in the dump table block, obtain a new block and initialize it with null values. Create a dump table entry in the next available entry, indicated by TDTFREEHEAD pointer in the anchor block, using the parameter values passed by the caller. Set up the NEXT and PREV pointers of the new entry and higher and lower entries to include the new entry in the correct sequence in the table.
3. Write the dump code information to the global catalog.

DUDT gate, DELETE_TRAN_DUMPCODE function

The DELETE_TRAN_DUMPCODE function of the DUDT gate is invoked to delete an existing dump code from the transaction dump table.

Input parameters

TRANSACTION_DUMPCODE
is the transaction dump code.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUMPCODE_NOT_FOUND IO_ERROR

Process flow

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT DELETE_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

DFHDUTM process flow

1. Locate the dump code in the transaction dump table. If it cannot be found, return to the caller indicating DUMPCODE_NOT_FOUND exception.
2. Adjust the NEXT and PREV of the higher and lower entries in the table to bypass this entry, and set its NEXT and PREV pointers to 0.

3. Delete the information for the dump code from the global catalog. If the attempt to delete from the catalog indicates that the record is not found, it is assumed that the dump code was present on the dump table as a result of a LOCATE_TRAN_DUMPCODE subroutine call that does not update the catalog.

DUDT LOCATE_TRAN_DUMPCODE process flow

1. Validate the dump code for which a dump has been requested (see ADD_TRAN_DUMPCODE).
2. Search the transaction dump table for the dump code. If it is found, set up the return DUDT parameters to indicate whether CICS is to be terminated, and whether a system or transaction dump is to be taken, using values taken from the dump table entry.

   If the dump code does not exist on the dump table, an entry is added, using default values (see the *CICS Problem Determination Guide*) and the DUDT return parameters are set up dependent on these default values. (This default entry is not added to the global catalog.)

DUDT gate, INQUIRE_TRAN_DUMPCODE function

The INQUIRE_TRAN_DUMPCODE function of the DUDT gate is invoked to inquire on a dump code in the transaction dump table.

Input parameters

TRANSACTION_DUMPCODE
is the transaction dump code.

Output parameters

[DUMPSCOPE]
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:

LOCAL|RELATED

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

[TRANSACTION_DUMP]
states whether a transaction dump is required for this dump code. It can have either of these values:

YES|NO

[SYSTEM_DUMP]
states whether a system dump is required for this dump code. It can have either of these values:

YES|NO

[TERMINATE_CICS]
states whether CICS is to be terminated for this dump code. It can have either of these values:

YES|NO

[MAXIMUM_DUMPS]
is the maximum number of times the dump code action can be taken.

[COUNT]
is the number of times the dump code action has been taken.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:

DUMPCODE_NOT_FOUND

Process flow

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT INQUIRE_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

DFHDUTM process flow

1. Locate the dump code in the transaction dump code table. If it cannot be found, return to the caller indicating DUMPCODE_NOT_FOUND exception.
2. Return the dump code table entry information to the caller in the DUDT parameters.

DUDT gate, SET_TRAN_DUMPCODE function

The SET_TRAN_DUMPCODE function of the DUDT gate is invoked to set options for a dump code in the transaction dump table.

Input parameters

[DUMPSCOPE]
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:

LOCAL|RELATED

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

TRANSACTION_DUMPCODE
is the transaction dump code.

[TRANSACTION_DUMP]
states whether a transaction dump is required for this dump code. It can have either of these values:

YES|NO

[SYSTEM_DUMP]
states whether a system dump is required for this dump code. It can have either of these values:

YES|NO

[TERMINATE_CICS]
states whether CICS is to be terminated for this dump code. It can have either of these values:

YES|NO

[MAXIMUM_DUMPS]
is the maximum number of times the dump code action can be taken.

[RESET_COUNT]
states whether COUNT is to be reset to zero. It can have either of these values:

YES|NO

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUMPCODE_NOT_FOUND CATALOG_FULL IO_ERROR

Process flow

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT SET_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

DFHDUTM process flow

1. Locate the dump code in the transaction dump code table. If it cannot be found, return to the caller indicating DUMPCODE_NOT_FOUND exception.
2. Change the values on the dump code table entry for any passed in the DUDT parameter list (some or all may be changed). If the RESET_COUNT parameter is present, set the count of the number of dumps taken for this dump code to zero.
3. Make the same changes to the dump code information about the global catalog. If the attempt to delete from the catalog indicates that the record is not found, it is assumed that the dump code was present on the dump table as a result of a LOCATE_TRAN_DUMPCODE subroutine call that does not update the catalog. See "DUDT LOCATE_TRAN_DUMPCODE process flow" on page 724 for a description of the process flow of this function.

DUDT gate, STARTBR_TRAN_DUMPCODE function

The STARTBR_TRAN_DUMPCODE function of the DUDT gate is invoked to start a browse session on the transaction dump table.

Input parameters

None.

Output parameters

BROWSE_TOKEN is the token identifying the browse session.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. It has this value:

INSUFFICIENT_STORAGE

Process flow

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT STARTBR_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

DFHDUTM process flow

1. Add a new browse token to the end of the browse token table. Set the value of the last dump code used to null in the browse token table entry.
2. Return the browse token to the caller.

DUDT gate, GETNEXT_TRAN_DUMPCODE function

The GETNEXT_TRAN_DUMPCODE function of the DUDT gate is invoked in a browse session to get the next entry in the transaction dump table.

Input parameters

BROWSE_TOKEN is the token identifying the browse session.

Output parameters

[DUMPSCOPE] indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:

LOCAL|RELATED

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

[TRANSACTION_DUMPCODE]
is the transaction dump code.

[TRANSACTION_DUMP]
states whether a transaction dump is required for this dump code. It can have either of these values:

YES|NO

[SYSTEM_DUMP]
states whether a system dump is required for this dump code. It can have either of these values:

YES|NO

[TERMINATE_CICS]
states whether CICS is to be terminated for this dump code. It can have either of these values:

YES|NO

[MAXIMUM_DUMPS]
is the maximum number of times the dump code action can be taken.

[COUNT] is the number of times the dump code action has been taken.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	END_BROWSE
INVALID	INVALID_BROWSE_TOKEN

Process flow

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT GETNEXT_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

DFHDUTM process flow

1. Search the browse token table for the browse token passed in the DUDT parameters. If the browse token cannot be found, perform error handling (exception trace, message, and dump) and return to the caller.
2. Obtain the value of the last dump code read by this browse session from the browse token table entry, and scan the dump table for a higher dump code entry. If there are no more entries, return END_BROWSE exception to the call; otherwise return the details of the dump code table entry in the parameters and save the value of the dump code in the browse token table entry.

DUDT gate, ENDBR_TRAN_DUMPCODE function

The ENDBR_TRAN_DUMPCODE function of the DUDT gate is invoked to end a browse session on the transaction dump table.

Input parameters

BROWSE_TOKEN
is the token identifying the browse session.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. It has this value:

INVALID_BROWSE_TOKEN

Process flow

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT ENDBR_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

DFHDUTM process flow

1. Search the browse token table for the browse token passed in the DUDT parameters. If the browse token cannot be found, perform error handling (exception trace, message, and dump) and return to the caller.
2. Set the browse token table entry to nulls and adjust the NEXT and PREV pointers to bypass the entry.

DUDT gate, ADD_SYSTEM_DUMPCODE function

The ADD_SYSTEM_DUMPCODE function of the DUDT gate is invoked to add a new dump code to the system dump table.

Input parameters

DAEOPTION
states whether a dump produced for this dumpcode is eligible for suppression by the MVS Dump Analysis and Elimination (DAE) component. It can have either of these values:

YES|NO

DUMPSCOPE
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:

LOCAL|RELATED

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

SYSTEM_DUMPCODE
is the system dump code.

SYSTEM_DUMP
states whether a system dump is required for this dump code. It can have either of these values:

YES|NO

TERMINATE_CICS
states whether CICS is to be terminated for this dump code. It can have either of these values:

YES|NO

MAXIMUM_DUMPS
is the maximum number of times the dump code action can be taken.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE_DUMPCODE INVALID_DUMPCODE CATALOG_FULL INSUFFICIENT_STORAGE IO_ERROR

Process flow

1. Acquire KE system dump lock.
2. Issue DUDT ADD_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

DFHDUTM process flow

1. Validate the dump code to be added. System dump codes are 4 bytes and must not contain leading or embedded blanks. If the dump code is not valid, return to the caller indicating the exception.
2. Scan the system dump table to find the correct place to insert the dump code in collating sequence. If an entry already exists for that dump code, return to the caller indicating duplicate dump code. If the entry is about to use the last available entry in the dump table block, obtain a new block and initialize it with null values. Create a dump table entry in the next available entry, indicated by TDTFREEHEAD pointer in the anchor block, using the parameter values passed by the caller. Set up the NEXT and PREV pointers of the new entry and higher and lower entries to include the new entry in the correct sequence in the table.
3. Write the dump code information to the global catalog.

DUDT gate, DELETE_SYSTEM_DUMPCODE function

The DELETE_SYSTEM_DUMPCODE function of the DUDT gate is invoked to delete an existing dump code from the system dump table.

Input parameters

SYSTEM_DUMPCODE
is the system dump code.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUMPCODE_NOT_FOUND IO_ERROR

Process flow

1. Acquire KE system dump lock.
2. Issue DUDT DELETE_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

DFHDUTM process flow

1. Locate the dump code in the system dump table. If it cannot be found, return to the caller indicating DUMPCODE_NOT_FOUND exception.
2. Adjust the NEXT and PREV of the higher and lower entries in the table to bypass this entry, and set its NEXT and PREV pointers to 0.
3. Delete the information for the dump table from the global catalog. If the attempt to delete from the catalog indicates that the record is not found, it is assumed that the dump code was present on the dump table as a result of a LOCATE_SYSTEM_DUMPCODE subroutine call that does not update the catalog.

DUDT LOCATE_SYSTEM_DUMPCODE process flow

1. Validate the dump code for which a dump has been requested (see ADD_SYSTEM_DUMPCODE).
2. Search the system dump table for the dump code. If it is found, set up the return DUDT parameters to indicate whether CICS is to be terminated, and whether a system dump is to be taken, using values taken from the dump table entry.

   If the dump code does not exist on the dump table, an entry is added, using default values (see the *CICS Problem Determination Guide*) and the DUDT return parameters are set up dependent on these default values. (This default entry is not added to the global catalog.)

DUDT gate, INQUIRE_SYSTEM_DUMPCODE function

The INQUIRE_SYSTEM_DUMPCODE function of the DUDT gate is invoked to inquire on a dump code in the system dump table.

Input parameters

SYSTEM_DUMPCODE
: is the system dump code.

Output parameters

DAEOPTION
: states whether a dump produced for this dumpcode is eligible for suppression by the MVS Dump Analysis and Elimination (DAE) component. It can have either of these values:

  YES|NO

DUMPSCOPE
: indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:

  LOCAL|RELATED

  LOCAL
  : indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

  RELATED
  : indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

[SYSTEM_DUMP]
: states whether a system dump is required for this dump code. It can have either of these values:

  YES|NO

[TERMINATE_CICS]
: states whether CICS is to be terminated for this dump code. It can have either of these values:

  YES|NO

[MAXIMUM_DUMPS]
: is the maximum number of times the dump code action can be taken.

[COUNT]
: is the number of times the dump code action has been taken.

RESPONSE
: is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. It has this value:

DUMPCODE_NOT_FOUND

Process flow

1. Acquire KE system dump lock.
2. Issue DUDT INQUIRE_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

DFHDUTM process flow

1. Locate the dump code in the system dump code table. If it cannot be found, return to the caller indicating DUMPCODE_NOT_FOUND exception.
2. Return the dump code table entry information to the caller in the DUDT parameters.

DUDT gate, SET_SYSTEM_DUMPCODE function

The SET_SYSTEM_DUMPCODE function of the DUDT gate is invoked to set options for a dump code in the system dump table.

Input parameters

[DAEOPTION] states whether a dump produced for this dumpcode is eligible for suppression by the MVS Dump Analysis and Elimination (DAE) component. It can have either of these values:

YES|NO

[DUMPSCOPE] indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:

LOCAL|RELATED

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

SYSTEM_DUMPCODE
is the system dump code.

[SYSTEM_DUMP]
states whether a system dump is required for this dump code. It can have either of these values:

YES|NO

[TERMINATE_CICS]
states whether CICS is to be terminated for this dump code. It can have either of these values:

YES|NO

[MAXIMUM_DUMPS]
is the maximum number of times the dump code action can be taken.

[RESET_COUNT]
states whether COUNT is to be reset to zero. It can have either of these values:

YES|NO

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUMPCODE_NOT_FOUND CATALOG_FULL IO_ERROR

Process flow

1. Acquire KE system dump lock.
2. Issue DUDT SET_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

DFHDUTM process flow

1. Locate the dump code in the system dump code table. If it cannot be found, return to the caller indicating DUMPCODE_NOT_FOUND exception.
2. Change the values on the dump code table entry for any passed in the DUDT parameter list (some or all may be changed). If the RESET_COUNT parameter is present, set the count of the number of dumps taken for this dump code to zero.
3. Make the same changes to the dump code information about the global catalog. If the attempt to delete from the catalog indicates that the record is not found, it is assumed that the dump code was present on the dump table as a result of a LOCATE_SYSTEM_DUMPCODE subroutine call that does not update the catalog. See “DUDT LOCATE_SYSTEM_DUMPCODE process flow” on page 730 for a description of the process flow of this function.

DUDT gate, STARTBR_SYSTEM_DUMPCODE function

The STARTBR_SYSTEM_DUMPCODE function of the DUDT gate is invoked to start a browse session on the system dump table.

Input parameters

None.

Output parameters

BROWSE_TOKEN is the token identifying the browse session.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. It has this value:

INSUFFICIENT_STORAGE

Process flow

1. Acquire KE system dump lock.
2. Issue DUDT STARTBR_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

DFHDUTM process flow

1. Add a new browse token to the end of the browse token table. Set the value of the last dump code used to null in the browse token table entry.
2. Return the browse token to the caller.

DUDT gate, GETNEXT_SYSTEM_DUMPCODE function

The GETNEXT_SYSTEM_DUMPCODE function of the DUDT gate is invoked in a browse session to get the next entry in the system dump table.

Input parameters

BROWSE_TOKEN is the token identifying the browse session.

Output parameters

[DAEOPTION]
states whether a dump produced for this dumpcode is eligible for suppression by the MVS Dump Analysis and Elimination (DAE) component. It can have either of these values:

YES|NO

[DUMPSCOPE]
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:

LOCAL|RELATED

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

[SYSTEM_DUMPCODE]
is the system dump code.

[SYSTEM_DUMP]
states whether a system dump is required for this dump code. It can have either of these values:

YES|NO

[TERMINATE_CICS]
states whether CICS is to be terminated for this dump code. It can have either of these values:

YES|NO

[MAXIMUM_DUMPS]
is the maximum number of times the dump code action can be taken.

[COUNT]
is the number of times the dump code action has been taken.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	END_BROWSE
INVALID	INVALID_BROWSE_TOKEN

Process flow

1. Acquire KE system dump lock.
2. Issue DUDT GETNEXT_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

DFHDUTM process flow

1. Search the browse token table for the browse token passed in the DUDT parameters. If the browse token cannot be found, perform error handling (exception trace, message, and dump) and return to the caller.
2. Obtain the value of the last dump code read by this browse session from the browse token table entry, and scan the dump table for a higher dump code entry. If there are no more entries, return END_BROWSE exception to the call; otherwise return the details of the dump code table entry in the parameters and save the value of the dump code in the browse token table entry.

DUDT gate, ENDBR_SYSTEM_DUMPCODE function

The ENDBR_SYSTEM_DUMPCODE function of the DUDT gate is invoked to end a browse on the system dump table.

Input parameters

BROWSE_TOKEN is the token identifying the browse session.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. It has this value:

INVALID_BROWSE_TOKEN

Process flow

1. Acquire KE system dump lock.
2. Issue DUDT ENDBR_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

DFHDUTM process flow

1. Search the browse token table for the browse token passed in the DUDT parameters. If the browse token cannot be found, perform error handling (exception trace, message, and dump) and return to the caller.
2. Set the browse token table entry to nulls and adjust the NEXT and PREV pointers to bypass the entry.

DUDU gate, TRANSACTION_DUMP function

The TRANSACTION_DUMP function of the DUDU gate is invoked to take a transaction dump.

Input parameters

Note: The [SEGMENT] and [SEGMENT_LIST] parameters are ***mutually exclusive***.

TRANSACTION_DUMPCODE
is a 4-character identifier for this dump request, used to index the transaction dump table to determine the options to be used.

The following set of optional input parameters indicates which parts of storage are to be included in the transaction dump. Each parameter can have either of these values: YES|NO.

[CSA] – common system area
[TCA] – task control area
[PROGRAM] – program storage
[TRT] – internal trace table
[TERMINAL] – terminal-related storage areas
[TRANSACTION]
– transaction-related storage areas
[SIT] – system initialization table
[PPT] – processing program table
[PCT] – program control table
[TCT] – terminal control table
[FCT] – file control table
[DCT] – destination control table.

[SEGMENT] specifies the address and length of a single block of storage to be dumped.

[SEGMENT_LIST]
specifies the address and length of a list of length-address pairs of storage blocks to be dumped. SEGMENT and SEGMENT_LIST may not be specified together.

[INDIRECT_CALL]
states whether the call is indirect, that is, whether the actual requester of the dump is not the immediate caller of the dump domain. It can have either of these values:

YES|NO

Output parameters

DUMPID
is a character string of the form "rrrr/cccc" giving a unique identification to this dump request. "rrrr" is the run number of this CICS instance. Leading zeros are removed. The run number is incremented every time CICS is initialized. "cccc" is the count of this dump request within this CICS run.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. It can have any one of these values:

FESTAE_FAILED
The MVS FESTAE macro failed to set up a functional recovery routine during the processing of the system dump request.

IWMWQWRK_FAILED
An MVS IWMWQWRK macro call to Workload Manager returned a warning or error response during the processing of the system dump request.

INVALID_SVC_CALL
DFHDUSVC received a request for an invalid function.

INVALID_PROBDESC
The MVS PROBDESC parameters, which CICS creates and passes to MVS on an SDUMP call, contained invalid data.

OPEN_ERROR
Failed to open the CICS dump data set during an autoswitch.

NOT_OPEN
The dump data set is currently closed.

INVALID_DUMPCODE
The transaction dump code failed validation.

PARTIAL_TRANSACTION_DUMP
There was insufficient space in the current dump data set for this dump. Autoswitching had not been requested.

SUPPRESSED_BY_DUMPOPTION
A system dump requested through the dump table for this transaction dump code was suppressed because the DUMP=NO system initialization parameter had been specified.

SUPPRESSED_BY_DUMPTABLE
The dump table specified that no dump was required for this dump code.

SUPPRESSED_BY_USEREXIT
The XDUREQ user exit requested suppression of this dump.

PARTIAL_SYSTEM_DUMP
A system dump requested through the dump table for this transaction dump code was incomplete because of insufficient space on the SYS1.DUMP data set.

SDUMP_FAILED
A system dump requested through the dump table for this transaction dump code failed because of an MVS or I/O failure.

SDUMP_BUSY
A system dump requested through the dump table for this system dump code failed because another address space was in the process of taking
an SDUMP or the task was cancelled using the KILL option.

SDUMP_NOT_AUTHORIZED
A system dump requested through the dump table for this transaction dump code failed because the CICS authorized function control block (AFCB) indicates that CICS use of SDUMP is not authorized.

INSUFFICIENT_STORAGE
A system dump requested through the dump table for this transaction dump code failed because CICS failed to acquire the necessary storage to build the SDUMP parameter list.

NO_DATASET
A system dump requested through the dump table for this transaction dump code failed because there were no SYS1.DUMP data sets available.

Process flow

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT LOCATE_TRAN_DUMPCODE call to DFHDUTM. If the dump table is not available, CICS takes a system dump and terminates.
3. Issue LMLM UNLOCK for DUTABLE lock.
4. If XDUREQ exit active, issue APEX INVOKE_USER_EXIT.
5. If XDUREQ exit not active or it was active and the return code was zero:
   - If dump table indicates that a system dump is required for this transaction dump code and the DUMP=NO system initialization parameter was not specified. invoke CICS SVC to take system dump, retrying as necessary if SDUMP is busy.
   - If dump table indicates that a transaction dump is required, call DFHDUXD with a DUDD format parameter list to take a transaction dump.
6. If XDUREQC exit active, issue APEX INVOKE_USER_EXIT.
7. Issue LMLM LOCK for DUTABLE lock.
8. Issue DUDT COMMIT_TRAN_DUMPCODE call to DFHDUTM.
9. Issue LMLM UNLOCK for DUTABLE lock.
10. Issue KEDD PERFORM_SYSTEM_ACTION to terminate CICS if the dump table indicated that termination was required for this dump code.

DUDD TAKE_DUMP process flow

In DFHDUXD:

1. If dump data set is closed or is a dummy data set, and the XDUOUT exit is not active, return to caller.
2. Issue LMLM LOCK for dump data set lock.
3. Invoke transaction dump formatting routines (DFHxxXDF), with DUXF FORMAT function, in turn to dump required areas to the transaction dump data set. If, at any point, the DUXF FORMAT function returns a response of EXCEPTION and a reason of RESTART, an autoswitch has occurred and the DUXF FORMAT calls have to be issued again.
4. Issue LMLM UNLOCK for dump data set lock.
5. If DFHDUXD is terminating with a DISASTER response and XDUOUT is active, issue APEX INVOKE_USER_EXIT for XDUOUT, passing the abnormal termination indication.

DUDT COMMIT_TRAN_DUMPCODE process flow

The DUDT COMMIT_TRAN_DUMPCODE function updates statistics for the dump code, according to whether or not the dump domain took the requested dumps.

1. Locate the entry on the transaction dump table. Return to the caller, indicating exception if the entry is not found.
2. Increment the global system dump statistics in the DUA and the system dump statistics on the dump table entry, for either dump-taken or dump-suppressed depending on the input system-dump parameter.
3. Increment the global transaction dump statistics in the DUA and the transaction dump statistics for either dump-taken or dump-suppressed depending on the input transaction-dump parameter.

DUDU gate, SYSTEM_DUMP function

The SYSTEM_DUMP function of the DUDU gate is invoked to take a system dump.

Input parameters

SYSTEM_DUMPCODE
: is an 8-character identifier for this dump request, used to index the system dump table to determine the options to be used.

[MESSAGE_TEXT]
: specifies the address and length of the message text associated with this system dump.

[TITLE] specifies the address and length of a title to be associated with this dump.

[CALLER] specifies the address and length of a character string to appear as the caller of this dump.

[SYMPTOM_RECORD]
specifies the address and length of the symptom record associated with this dump.

[SYMPTOM_STRING]
specifies the address and length of the symptom string associated with this dump.

[TERMINATE_CICS]
states whether CICS is to be terminated after the dump if there is no entry in the dump table for this dump code; that is, it overrides the termination default of NO. It can have either of these values:

YES|NO

[INDIRECT_CALL]
states whether the call is indirect, that is, whether the actual requester of the dump is not the immediate caller of the dump domain. It can have either of these values:

YES|NO

Output parameters

DUMPID is a character string of the form “rrrr/cccc” giving a unique identification to this dump request. “rrrr” is the run number of this CICS instance. Leading zeros are removed. The run number is incremented every time CICS is initialized. “cccc” is the count of this dump request within this CICS run.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. It can have any one of these values:

INVALID_DUMPCODE
The system dump code failed validation.

SUPPRESSED_BY_DUMPOPTION
A system dump requested through the dump table for this system dump code was suppressed because the DUMP=NO system initialization parameter had been specified.

SUPPRESSED_BY_DUMPTABLE
The dump table specified that no dump was required for this dump code.

SUPPRESSED_BY_USEREXIT
The XDUREQ user exit requested suppression of this dump.

PARTIAL_SYSTEM_DUMP
A system dump requested through the dump table for this system dump code was incomplete because of insufficient space on the SYS1.DUMP data set.

SDUMP_FAILED A system dump requested through the dump table for this system dump code failed because of an MVS or I/O failure.

SDUMP_BUSY A system dump requested through the dump table for this system dump code failed because another address space was in the process of taking an SDUMP or the task was cancelled using the KILL option.

SDUMP_NOT_AUTHORIZED
A system dump requested through the dump table for this system dump code failed because the CICS authorized function control block (AFCB) indicates that CICS use of SDUMP is not authorized.

INSUFFICIENT_STORAGE
A system dump requested through the dump table for this system dump code failed because CICS failed to acquire the necessary storage to build the SDUMP parameter list.

NO_DATASET A system dump requested through the dump table for this system dump code failed because there were no SYS1.DUMP data sets available.

Process flow

1. Acquire KE system dump lock.
2. If the DUMP=YES system initialization parameter was specified:
   - Issue DUDT LOCATE_SYSTEM_DUMPCODE call to DFHDUTM.
   - If dump table indicates system dump required:
     - If XDUREQ exit active, issue APEX INVOKE_USER_EXIT.
     - If XDUREQ exit not active or it was active and the return code was zero, invoke CICS SVC to take system dump, retrying as necessary if SDUMP is busy.
     - If XDUREQC exit active, issue APEX INVOKE_USER_EXIT.
3. Issue DUDT COMMIT_SYSTEM_DUMPCODE call to DFHDUTM.
4. Release KE system dump lock.
5. Issue KEDD PERFORM_SYSTEM_ACTION to terminate CICS if the dump table indicated that termination was required for this dump code.

DUDT COMMIT_SYSTEM_DUMPCODE process flow

The COMMIT_SYSTEM_DUMPCODE function of the DUDT gate updates statistics for the dump code, according to whether or not the dump domain took the requested dumps.

- Locate the entry on the system dump table. Return to the caller, indicating exception if the entry is not found.
- Increment the global system dump statistics and the system dump statistics on the dump table entry, for either dump-taken or dump-suppressed depending on the input system-dump parameter.

DUSR gate, CROSS_SYSTEM_DUMP_AVAIL function

The CROSS_SYSTEM_DUMP_AVAIL function of the DUSR gate is used to inform the dump domain about the DUMP_AVAIL token which links CICS with the MVS workload manager.

Input parameters

CROSS_SYSTEM_DUMP_AVAIL
is the CICS to MVS workload manager token.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|DISASTER|INVALID|KERNERROR|PURGED`

Process flow

- Set the CICS to MVS workload manager connect token in the DUA.

DUSR gate, DUMPDS_OPEN function

The DUMPDS_OPEN function of the DUSR gate is invoked to open the CICS dump data set.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:
OPEN_ERROR

which indicates that the MVS OPEN of the dump data set failed.

Process flow

1. Issue LMLM LOCK for dump data set lock.
2. Call DUSU OPEN function.
3. Issue LMLM UNLOCK for dump data set lock.

DUSR gate, DUMPDS_CLOSE function

The DUMPDS_CLOSE function of the DUSR gate is invoked to close the CICS dump data set.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

Process flow

1. Issue LMLM LOCK for dump data set lock.
2. Call DUSU CLOSE function.
3. Issue LMLM UNLOCK for dump data set lock.

DUSR gate, DUMPDS_SWITCH function

The DUMPDS_SWITCH function of the DUSR gate is invoked to switch to the alternate CICS dump data set.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:
OPEN_ERROR

which indicates that the MVS OPEN of the dump data set failed.

Process flow

1. Issue LMLM LOCK for dump data set lock.
2. Call DUSU SWITCH function.
3. Issue LMLM UNLOCK for dump data set lock.

DUSU SWITCH process flow

1. Process as for DUSU CLOSE.
2. Switch current data set name in the DUA.
3. Process as for DUSU OPEN.

DUSU OPEN process flow

1. Return if the DUA indicates already open.
2. Call DUIO OPEN function.
3. Update status on catalog.

DUSU CLOSE process flow

1. If data set is open:
   - Call DUIO ALLOC_STG function to get storage for DYNALLOC parameter list.
   - Issue DYNALLOC to get data set name for current dump data set.
2. Call DUIO CLOSE function.
3. If XDUCLSE exit is active, call APEX INVOKE_USER_EXIT.
4. Set status in the DUA to closed.
5. Free DYNALLOC parameter list if necessary.

DUIO OPEN process flow

1. Return if the DUA indicates transaction dump data set is already open.
2. Issue MVS GETMAIN for DU Open Block if it is not yet allocated.
3. Issue MVS OPEN.
4. Set status to open in the DUA.
5. Write end-of-data record.

DUIO uses the DCB OPEN exit to complete the DCB with block size and LRECL, and to determine the size of the buffer to be used by CICS. The DCB abend exit and the SYNAD routine are also activated to detect any errors that may occur during OPEN.

DUIO CLOSE process flow

1. Return if already closed.
2. Issue MVS CLOSE.
3. Issue MVS FREEPOOL to release buffers.
4. If this close is not for a switch, free the DU open block.
5. Set status to closed in the DUA.

DUIO ALLOC_STG process flow

1. Issue MVS GETMAIN for requested storage.
2. Clear acquired area to hexadecimal zeros.

DUSR gate, INQUIRE_CURRENT_DUMPDS function

The INQUIRE_CURRENT_DUMPDS function of the DUSR gate returns the name of the current dump data set.

Input parameters

None.

Output parameters

CURRENT_DUMPDS
is the name of the current dump data set. It can have either of these values:

DFHDMPA|DFHDMPB

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

DUSR gate, INQUIRE_DUMPDS_OPEN_STATUS function

The INQUIRE_DUMPDS_OPEN_STATUS function of the DUSR gate returns an indication of whether the current dump data set is open or closed.

Input parameters

None.

Output parameters

OPEN_STATUS
is the open status of the current dump data set. It can have either of these values:

OPEN|CLOSED

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

DUSR gate, INQUIRE_DUMPDS_AUTOSWITCH function

The INQUIRE_DUMPDS_AUTOSWITCH function of the DUSR gate returns an indication of whether autoswitching is active or not.

Input parameters

None.

Output parameters

AUTOSWITCH
is the dump data set autoswitch status. It can have either of these values:

ON|OFF

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

DUSR gate, SET_DUMPDS_AUTOSWITCH function

The SET_DUMPDS_AUTOSWITCH function of the DUSR gate is used to set autoswitching on or off.

Input parameters

AUTOSWITCH
is the dump data set autoswitch status. It can have either of these values:

ON|OFF

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

Process flow

1. Set new autoswitch value in the DUA.
2. Call DUSU UPDATE_CATALOGUE function, to write the DU state record to local catalog, using the current status from the DUA.

DUSR gate, INQUIRE_INITIAL_DUMPDS function

The INQUIRE_INITIAL_DUMPDS function of the DUSR gate returns the setting of the initial dump data set.

Input parameters

None.

Output parameters

INITIAL_DUMPDS
is the initial dump data set. It can have any one of these values:

DFHDMPA
Open DFHDMPA first when CICS is next initialized.

DFHDMPB
Open DFHDMPB first when CICS is next initialized.

AUTO
When CICS is next initialized, open the extent that was not active when CICS last terminated.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

DUSR gate, SET_INITIAL_DUMPDS function

The SET_INITIAL_DUMPDS function of the DUSR gate is used to change the setting of the initial dump data set.

Input parameters

INITIAL_DUMPDS
is the initial dump data set. It can have any one of these values:

DFHDMPA
Open DFHDMPA first when CICS is next initialized.

DFHDMPB
Open DFHDMPB first when CICS is next initialized.

AUTO
When CICS is next initialized, open the extent that was not active when CICS last terminated.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

Process flow

1. Set new initial dump data set value in the DUA.
2. Call DUSU UPDATE_CATALOGUE function, to write the DU state record to local catalog, using the current status from the DUA.

DUSR gate, SET_DUMPTABLE_DEFAULTS function

The SET_DUMPTABLE_DEFAULTS function of the DUSR gate is invoked during system initialization tp update the DUA with the DAE option specified in a SIT or as a SIT override.

Input parameters

DAE_DEFAULT
indicates whether temporary dump table entries added by CICS will indicate DAE (dump eligible for DAE suppression) of NODAE (dump will not be suppressed by DAE). It can have either of the values:

DAE|NODAE

SYDUMAX_DEFAULT
is taken from system initialization parameter (SIT=SYDUMAX), which specifies the maximum number of system dumps which can be taken per dump table entry.

TRDUMAX_DEFAULT
is taken from system initialization parameter (SIT=TRDUMAX), which specifies the maximum number of transaction dumps which can be taken per dump table entry.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

Process flow

1. Set DAE_DEFAULT flag value in the DUA. 1 indicates DAE, 0 indicates NODAE.
2. Call DUSU UPDATE_CATALOGUE function, to write the DU state record to local catalog, using the current status from the DUA.

DUSR gate, INQUIRE_SYSTEM_DUMP function

The INQUIRE_SYSTEM_DUMP function of the DUSR gate returns the setting of the system dump suppression flag.

Input parameters

None.

Output parameters

SYSTEM_DUMP is the system dump option, indicating whether or not SDUMPs are to be taken by this CICS system. It can have either of these values:

YES|NO

where NO means that SDUMPs are not taken by this CICS system.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

DUSR gate, SET_SYSTEM_DUMP function

The SET_SYSTEM_DUMP function of the DUSR gate is used to change the setting of the system dump suppression flag.

Input parameters

SYSTEM_DUMP is the system dump option, indicating whether or not SDUMPs are to be taken by this CICS system. It can have either of these values:

YES|NO

where NO means that SDUMPs are not taken by this CICS system.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

Process flow

- Set new system dump suppression flag value in the DUA.
- Call DUSU UPDATE_CATALOGUE function, to write the DU state record to local catalog, using the current status from the DUA.

DUSR gate, INQUIRE_RETRY_TIME function

The INQUIRE_RETRY_TIME function of the DUSR gate returns the value of the SDUMP retry time.

Input parameters

None.

Output parameters

RETRY_TIME is the value in seconds of the time interval for which CICS should retry SDUMP requests that fail because another SDUMP is in progress within the MVS system. The SDUMP is retried at intervals of five seconds for the specified total time.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

DUSR gate, SET_RETRY_TIME function

The SET_RETRY_TIME function of the DUSR gate is invoked to set the SDUMP retry time.

Input parameters

RETRY_TIME
is the value in seconds of the time interval for which CICS should retry SDUMP requests that fail because another SDUMP is in progress within the MVS system. The SDUMP is retried at intervals of five seconds for the specified total time.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

Process flow

1. Set new SDUMP retry time in the DUA.
2. Call DUSU UPDATE_CATALOGUE function, to write the DU state record to local catalog, using the current status from the DUA.

Miscellaneous process flows

DUIO format, WRITE function

1. If the first record in the block to be written is a dump header:
   - Issue MVS NOTE to get location of last record written (an end-of-data record).
   - Issue MVS POINT to position for overwrite of the end-of-data record.
2. Issue MVS WRITE.
3. Issue DSSR WAIT_MVS on the I/O ECB.
4. Issue MVS CHECK for I/O completion. This drives the DCB abend exit if an error or end-of-extent is encountered and results in an error or END_OF_EXTENT response from DUIO.

DUSU format, WRITE function

1. Call DUIO WRITE function if the dump data set is open and is not a dummy.
2. If an end-of-extent occurred:
   - If autoswitch is not active, close data set as for DUSU CLOSE above.
   - If autoswitch is active, turn autoswitch off and process as for DUSU SWITCH.

DUXF format, FORMAT function

This is the format of the parameter list passed to the transaction dump formatting routines (DFHxxXDF). There is a SUB_FUNCTION parameter which indicates the areas to be dumped. Each formatting routine is responsible for handling a subset of the subfunctions. The subfunctions and corresponding formatting modules are listed below in the order of the subfunction invocation from DFHDUXD.

Module	Subfunction
DFHXDXDF	DUXF_FORMAT_DUMP_HEADER
DFHXDXDF	DUXF_FORMAT_SHORT_SYMPTOM_STRIN
DFHXDXDF	DUXF_FORMAT_CICS_SERVICE_LEVEL
DFHXDXDF	DUXF_FORMAT_PSW_REGISTERS
DFHSAXDF	DUXF_FORMAT_TCA
DFHPCXDF	DUXF_FORMAT_LIFO
DFHSAXDF	DUXF_FORMAT_COMM_AREAS
DFHSAXDF	DUXF_FORMAT_CSA
DFHTRXDF	DUXF_FORMAT_TRT
DFHXDXDF	DUXF_FORMAT_SEGMENT

Module	Subfunction
DFHXDXDF	DUXF_FORMAT_SEGMENT_LIST
DFHSAXDF	DUXF_FORMAT_TRANSACTION_STORAGE
DFHSAXDF	DUXF_FORMAT_FCA
DFHTCXDF	DUXF_FORMAT_TCTTE
DFHPCXDF	DUXF_FORMAT_PROGRAM
DFHSAXDF	DUXF_FORMAT_DCT
DFHFCXDF	DUXF_FORMAT_FCT
DFHTCXDF	DUXF_FORMAT_TCT
DFHXRXDF	DUXF_FORMAT_XRF
DFHPCXDF	DUXF_FORMAT_PCT
DFHPCXDF	DUXF_FORMAT_PPT
DFHSAXDF	DUXF_FORMAT_SIT
DFHDLXDF	DUXF_FORMAT_DLI
DFHPCXDF	DUXF_FORMAT_MODULE_INDEX
DFHXDXDF	DUXF_FORMAT_DUMP_TRAILER

DUXW format, HEX function

1. Construct record in buffer indicating that this data should be formatted as hexadecimal.
2. If buffer is full, call DUSU WRITE to output it.
3. If XDUOUT exit is active, call APEX INVOKE_USER_EXIT.

DUXW format, NON_HEX function

1. Construct record in buffer indicating that this data should be printed as-is; that is, it is already a character string.
2. If buffer is full, call DUSU WRITE to output it.
3. If XDUOUT exit is active, call APEX INVOKE_USER_EXIT.

Dump domain's generic gates

Table 50 summarizes the dump domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 50. Dump domain's generic gates

Gate	Trace	Function	Format
DMDM	DU 0001 DU 0002	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
APUE	DU 0301 DU 0302	SET_EXIT_STATUS	APUE
STST	DU 0500 DU 0501	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format APUE—"Application domain's generic formats" on page 593

Format DMDM—"Domain manager domain's generic formats" on page 669

Format STST—"Statistics domain's generic format" on page 1198

In preinitialization processing, the dump domain establishes the initial dumping status:
- System dumping is enabled or suppressed, as required.
- The next transaction dump data set to be used is flagged.
- The transaction dump data set autoswitch status is set on or off, as required.
- The dump retry interval is established.
- The system dump table is initialized to empty.

For a cold start, the information comes from the system initialization parameters; for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

In initialization processing, the dump domain loads the transaction dump table and the system dump table from the global catalog.

In quiesce processing, the dump domain performs only internal routines.

In termination processing, the dump domain closes the transaction dump data set.

DMDM PRE_INITIALIZE function

The PRE_INITIALIZE function of the DMDM gate performs the following functions:
1. Issue MVS GETMAIN for DU anchor block (DUA) and initialize it.
2. Read DU state record from the local catalog and set values in the DUA.
3. Initialize to empty the system dump table.
4. Issue MVS GETMAIN for DU statistics buffer.
5. Acquire startup information from the parameter manager (PA) domain and set it in the DUA.
6. Inform the kernel that DU system dump is available by issuing KEDD ADD_GATE for the DFHDUDU gate.

DMDM INITIALIZE_DOMAIN function

The INITIALIZE_DOMAIN function of the DMDM gate performs the following functions:
1. Load the system dump table from the global catalog.
2. Load the transaction dump table from the global catalog.
3. Issue LMLM ADD_LOCK for the dump data set lock (DUDATSET).
4. Issue LMLM ADD_LOCK for the dump table lock (DUTABLE).
5. Issue LMLM UNLOCK for DUTABLE lock.
6. Issue KEDD ADD_GATE for the DU STST, DUDT, and APUE gates.
7. Initialize transaction dump, including loading DFHDUIO, and indicate that the dump table is available to the DUDU TRANSACTION_DUMP function.
8. Update DU state record on catalog.
9. Issue LMLM UNLOCK for DUDATSET lock, thereby making the transaction dump function available.

DMDM QUIESCE_DOMAIN function

The QUIESCE_DOMAIN function of the DMDM gate issues a DMDM WAIT_PHASE function request to ensure all statistics are collected.

DMDM TERMINATE_DOMAIN function

The TERMINATE_DOMAIN function of the DMDM gate issues a DUSU CLOSE request to close the transaction dump data set.

APUE SET_EXIT_STATUS function

The SET_EXIT_STATUS function of the APUE gate sets the exit status flag in the DUA for the specified exit.

STST COLLECT_STATISTICS function

The COLLECT_STATISTICS function of the STST gate is called from the statistics domain. The process flow is:

1. Issue LMLM LOCK for DUTABLE lock on the transaction dump table.
2. Acquire KE system dump lock.
3. Issue STST COLLECT_STATISTICS call to DFHDUTM.
4. Release DUTABLE lock and system dump lock.

DFHDUTM process flow

If the COLLECT_STATISTICS parameters requested DATA, the following statistics records are written to the statistics domain:

1. If the RESOURCE_TYPE is not specified or is SYSDUMP, a DFHSDGPS global system dump statistics record is created, using global system dump counts (taken and suppressed) from the DUA. The KE system lock is released while a STATS_PUT request is made to the statistics domain. The lock is obtained again on successful completion of the STATS_PUT.
2. If the RESOURCE_TYPE is not specified or is TRANDUMP, a DFHTDGPS global transaction dump statistics record is created, using global transaction dump counts (taken and suppressed) from the DUA. The DUTABLE lock is released while a RECORD_STATISTICS request is made to the statistics domain. The lock is obtained again on successful completion of the RECORD_STATISTICS.
3. If the RESOURCE_TYPE is not specified or is SYSDUMP, a DFHSDRPS statistics detail record is written for every dump code found on the system dump table. The records contain the statistics for that dump code held on the dump table entry. The DFHSDRPS records are buffered and full buffers are written out using a RECORD_STATISTICS call to the statistics domain.
4. If the RESOURCE_TYPE is not specified or is TRANDUMP, a DFHTDRPS statistics detail record is written for every dump code found on the transaction dump table. The records contain the statistics for that dump code held on the dump table entry. The DFHTDRPS records are buffered and full buffers are written out using a RECORD_STATISTICS call to the statistics domain.

The global system and transaction dump counts (taken and suppressed) in the DUA are also reset to zero. The last_reset_time is also updated in the DUA at this time.

STST COLLECT_RESOURCE_STATS function

The COLLECT_RESOURCE_STATS function of the STST gate is called from an EXEC CICS command. The process flow is:

1. Issue LMLM LOCK for DUTABLE lock on the transaction dump table.
2. Acquire KE system dump lock.
3. Issue STST COLLECT_RESOURCE_STATS call to DFHDUTM.
4. Release DUTABLE lock and system dump lock.

DFHDUTM process flow

1. Validate RESOURCE_TYPE for either SYSDUMP or TRANDUMP. Perform error processing and return INVALID to the caller if it is neither of these.
2. If the RESOURCE_ID has not been passed, format a global statistics record, using counts of dumps taken and suppressed from the DUA, for either system or transaction dumps, depending on the RESOURCE_TYPE. Return this record to the caller in the RESOURCE_STATISTICS_DATA parameter.
3. If the RESOURCE_ID is present, it should contain a dump code. Search the relevant dump table (depending on RESOURCE_TYPE). Return ID_NOT_FOUND exception to the caller if the dump code

cannot be found. If the dump code is found, format either a DFHTDRPS or a DFHSDRPS statistics record using the dumps taken and suppressed statistics on the dump table entry. This record is formatted in the next available space in the RESOURCE_STATISTICS_DATA buffer.

Control blocks

Dump domain anchor block (DUA)
: There is one DU anchor block in the system. It is created when DU is initialized, and lasts for the lifetime of the system. It contains information relating to the status of the domain, and pointers to other control blocks.

Dump domain open block.
: This contains the data areas associated with the dump data set DCB, namely the ECB, DCB itself, DECB, and the output buffer. It resides below the 16MB line. It is allocated when the data set is opened, and freed when either an explicit close is issued or the end of the current data set is reached and autoswitching is not active.

System dump table (SDT)
: Storage for this table is obtained during dump domain preinitialization. The table is then initialized with null table entries. During dump domain initialization, the table is loaded with any values held on the global catalog for system dump codes that were explicitly added during previous CICS runs. Any system dumps taken before this point in initialization use default dump values (see the *CICS Problem Determination Guide* for information held for each dump code, and the default values).

Table entries are added during a CICS run either explicitly via CEMT or EXEC CICS commands, or implicitly, with default values, if a dump is requested for which an entry does not exist. These entries can be changed or deleted via CEMT or EXEC CICS commands. Explicitly added entries are written to the global catalog. Further blocks of storage are obtained if necessary as each block fills up. Storage for deleted entries is not reused, because activity on the table is low.

The DU domain anchor block contains pointers to the table, to the first and last active entries in the table, and to the next available entry. The table contains forward and backward pointers so that the table can be accessed in dump code sequence, and additional blocks are chained off the header of the previous block.

Transaction dump table (TDT)
: Storage for this table is obtained during dump domain initialization and the table is then loaded with any values held on the global catalog which were explicitly added during previous CICS runs.

Table entries are added during a CICS run either explicitly via CEMT or EXEC CICS commands, or implicitly, with default values, if a dump is requested for which an entry does not exist. These entries can be changed or deleted via CEMT or EXEC CICS commands. Explicitly added entries are written to the global catalog. Further blocks of storage are obtained if necessary as each block fills up. Storage for deleted entries is not reused, because activity on the table is low.

The DU domain anchor (DUA) block contains pointers to the table, to the first and last active entries in the table, and to the next available entry. The table contains forward and backward pointers so that the table can be accessed in dump code sequence, and additional blocks are chained off the header of the previous block.

Browse token table (BTT)
: This table holds browse tokens for both system and transaction dump tables. Each browse session started on either dump table is allocated a token that is held in this table, along with the dump code of the last dump table entry obtained by the browse session.

Storage for this table is obtained when the first dump table browse session of a CICS run is started. More storage is obtained when the table is full. Storage for deleted entries is not reused.

The structure of the table is the same as for the dump tables, as shown in Figure 109.

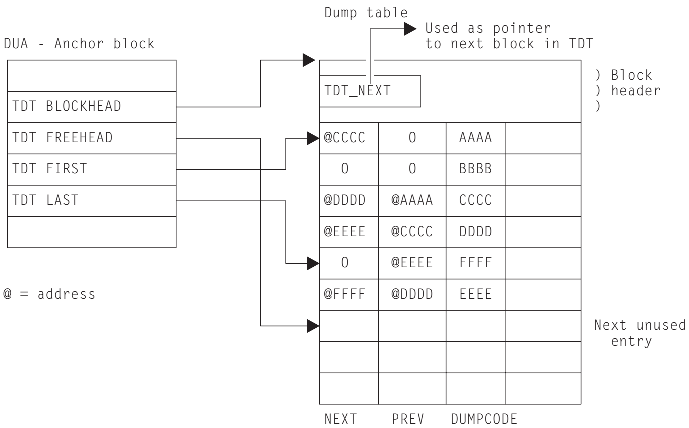


Figure 109. Format of the system and transaction dump tables and browse token table

Notes:

1. This example is for the transaction dump table, but it also applies to the SDT and the BTT.
2. The global catalog contained records for dump codes AAAA, BBBB, CCCC, DDDD, and FFFF.
3. Dump code BBBB has been deleted by an EXEC CICS command, so the NEXT and PREV pointers have been set to zero.
4. Dump code EEEE has been added during this CICS run and the pointers in entries for DDDD and FFFF adjusted to include EEEE in the correct sequence.
5. In this example, the first table block is not full, so TDT_NEXT in the block header is zero.

For a detailed description of these control blocks, see the *CICS Data Areas* manual.

Modules

Module	Function
DFHAPTRV	System dump formatting program, ZC Install
DFHAPTRY	System dump formatting program, XM related
DFHAPTRX	System dump formatting program, ZC persistent sessions
DFHDUDM	Processes requests to the DMDM gate of the dump domain
DFHDUDT	Processes requests to the DUDT gate of the dump domain
DFHDUDU	Processes requests to the DUDU gate of the dump domain
DFHDUIO	Processes domain subroutine requests of format DUIO
DFHDUSR	Processes requests to the DUSR and APUE gates of the dump domain
DFHDUSU	Processes domain subroutine requests of format DUSU
DFHDUSVC	System dump
DFHDUTM	Dump table manager
DFHDUXD	Invoked by DFHDUDU with a DUDD format parameter list to control the transaction dump process
DFHDUXW	Processes domain subroutine requests of format DUXW

Transaction dump formatting routines

The following routines are invoked by DFHDUXD to dump the storage areas associated with a particular CICS component. They are passed a DUXF format parameter list. They are all part of the DFHSIP load module.

Routine	Function
DFHDLXDF	DL/I related areas
DFHFCXDF	File control related areas
DFHPCXDF	Program related areas
DFHSAXDF	Common areas such as CSA, TCA, and so on
DFHSMXDF	Task subpools
DFHTCXDF	Terminal control related areas
DFHTRXDF	The internal trace table
DFHXDXDF	Information such as register contents, headers, and so on
DFHXRXDF	XRF related areas.

Copy books

Copy book	Function
DFHDUDCC	Contains the definitions of all DU control blocks.
DFHDUXDC	Provides common definitions for the transaction dump formatting routines DFHxxXDF.
DFHDUXDS	Common routine for the transaction dump formatting routines to convert responses from DFHDUXW into responses for DFHDUXD.
DFHDUXDV	Common abend recovery routine for the transaction dump formatting routines.

Exits

The dump domain exits are listed below. See the *CICS Customization Guide* for details of each exit.

XDUREQ
The dump request exit, driven for each transaction and system dump request.

XDUREQ
The dump request close exit, driven after a transaction or system dump has been taken (or failed or supressed).

XDUOUT
The output exit, driven before each buffer is written to the transaction dump data set.

XDUCLSE
The dump data set close exit, driven after each close of a transaction dump data set.

Trace

The point IDs for the dump domain are of the form DU xxxx; the corresponding trace levels are DU 1, DU 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Dumps

A formatted system dump contains the DU anchor (DUA) block and the DU open block.

System dumps requested by DU fall into two categories:

Dump code DUnnnn
For these there is a preceding console message DFHDUnnnn. See the *CICS Messages and Codes* manual for details.

Dump code KERNDUMP
If an error occurs in DFHDUDM during PRE_INITIALIZE processing, that is, before the system dump function is available, DU uses MVS WTO to write message DFHDU0103, and the kernel dump function to take an SDUMP.

Chapter 82. Enterprise Java domain (EJ)

The Enterprise Java (EJ) domain is logically divided into three parts:

- Elements, which covers the manipulation of the EJ Resources of CorbaServers (EJCG), DJars (EJDG) and Beans (EJBJ)
- Object Stores, used to store stateful Session Beans, and to hold the EJB Directory (EJOS and EJOB)
- Directory, used to record the association of OTS transactions and object instances with Request Processors (EJDI).

EJ domain's specific gates

Table 51 summarizes the EJ domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 51. EJ domain's specific gates

Gate	Trace	Function	XPI
EJBB	EJ 0Cxx	START_BROWSE GET_NEXT END_BROWSE	NO NO NO
EJBG	EJ 0Bxx	ADD_BEAN ADD_BEAN_STATS CONFIRM_ALL_BEANS DELETE_ALL_BEANS DELETE_BEAN GET_BEAN_DD INQUIRE_BEAN RESET_BEAN_STATS	NO NO NO NO NO NO NO NO
EJCB	EJ 08xx	START_BROWSE GET_NEXT END_BROWSE	NO NO NO
EJCG	EJ 07xx EJ 17xx	ADD_CORBASERVER ACTION_CORBASERVER AMEND_CORBASERVER DELETE_CORBASERVER ESTABLISH INQUIRE_CORBASERVER RELINQUISH RESOLVE_CORBASERVER SET_ALL_STATE WAIT_FOR_CORBASERVER	NO NO NO NO NO NO NO NO NO NO
EJDB	EJ 0Axx	START_BROWSE GET_NEXT END_BROWSE	NO NO NO
EJDG	EJ 09xx EJ 19xx	ADD_DJAR ACTION_DJAR AMEND_DJAR COUNT_FOR_CS DELETE_ALL_DJARS DELETE_DJAR INQUIRE_DJAR RESOLVE_DJAR SET_ALL_STATE WAIT_FOR_DJAR WAIT_FOR_USABLE_DJARS	NO NO NO NO NO NO NO NO NO NO NO
EJDI	EJ 05xx	ADD_ENTRY INITIALIZE LOOKUP_ENTRY REMOVE_ENTRY	NO NO NO NO
EJDU	EJ 06xx	DUMP_DATA DUMP_STACK INQUIRE_TRACE_FLAGS	NO NO NO

Table 51. EJ domain's specific gates (continued)

Gate	Trace	Function	XPI
EJGE	EJ 0Dxx	INITIALIZE QUIESCE TERMINATE	NO NO NO
EJIO	EJ 0Fxx	RESOLVE RESOLVE_CSERVERS RESOLVE_DJARS	NO NO NO
EJJO	EJ 0Exx	ADD_BEAN END_BEAN_BROWSE GET_BEAN_DD GET_NEXT_BEAN INQUIRE_CORBASERVER START_BEAN_BROWSE WAIT_FOR_CORBASERVER WAIT_FOR_USABLE_DJARS	NO NO NO NO NO NO NO NO
EJMI	EJ 50xx	ADD_BEAN ADD_METHOD DISCARD_METHOD_INFO GET_METHOD_INFO INITIALIZE	NO NO NO NO NO
EJOB	EJ 03xx	END_BROWSE_OBJECT GET_NEXT_OBJECSTORE INQUIRE_OBJECT INQUIRE_STORES RETRIEVE_STATISTICS START_BROWSE_OBJECT	NO NO NO NO NO NO
EJOS	EJ 02xx	ACTIVATE_OBJECT CLOSE_OBJECT_STORE OPEN_OBJECT_STORE REMOVE_OBJECT REMOVE_STORE STORE_OBJECT	NO NO NO NO NO NO
EJSO	EJ 1751 EJ 1752	INQUIRE_CORBASERVER AMEND_CORBASERVER	NO NO

EJBB gate, START_BROWSE Function

The START_BROWSE function of the EJBB gate initiates the browse upon the chain of Beans. Positioning of the start of the Browse is not supported. Selection by Bean is not provided, but selection by owning CorbaServer and owning DJar is. The end_browse condition is not returned if there are no suitable Beans (this is postponed until the get_next). The returned browsetoken must be used for subsequent GET_NEXT operations. This operation is available from EJJO and so the definitions must be consistent. The browsemode parameter controls which Beans are selected. Only BROWSEMODE(VALIDONLY) should be used by the SPI-layers.

- BROWSEMODE(ALL) selects all Beans (setting not usually used)
- BROWSEMODE(VALIDONLY) selects the Beans whose status has been confirmed (those which are not temporarily present during the install of all the Beans from a DJar). This is the usual (and default) setting.
- BROWSEMODE(INDOUBTONLY) selects the Beans whose status is temporary (those which are temporarily present during the install of all the Beans from a DJar).

Input parameters

CORBASERVER Name of the CorbaServer to be Browsed

DJAR Name of the DJar for this Bean

[BROWSEMODE] Controls which Beans are to be selected for Bean Browse

Output parameters

BROWSETOKEN The pointer set up by START_BROWSE which points to the first DJar in the chain to be Browsed

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	EJB_INACTIVE INVALID_BROWSEMODE INVALID_CORBASERVER INVALID_DJAR LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR

EJBB gate, GET_NEXT Function

The GET_NEXT function of the EJBB gate returns the next Bean Control Block in the list of Beans that meets the selection criteria. The ordering of Beans returned is not specified (the order is not alpha order but LastIn-FirstOut for Browse purposes). This operation is available from EJJO and so the definitions must be consistent. The POINTAT parameter is used to enable a Browse to proceed when the aim of the Browse is to locate a Bean to be deleted.

- POINTAT(NORMAL) should be used in all cases by the SPI layers and general users (and is the default).
- POINTAT(PRIOR) shows the deletion intent. POINTAT(PRIOR) should never be coded in normal circumstances and may result in an infinite loop if used without a delete.

Input parameters

BROWSETOKEN The pointer set up by START_BROWSE which points to the first DJar in the chain to be browsed

[POINTAT] Indicates whether to advance the browse pointer to point to the next item in the chain (NORMAL|PRIOR). NORMAL will return the next item in the chain, whereas PRIOR will always return the same item, unless that item has been deleted

Output parameters

BEAN Name of the Bean

[CORBASERVER]
Name of the CorbaServer for this DJar

[DDLEN] Length of the deployment/meta data area. Used particularly to contain the length of the data if the size is larger than the maximum length for ddareaforin block

[DJAR] Name of DJar for this Bean

[STATUS] The state of the Bean being Browsed (NORMAL or TEMPORARY). Indicates that a Bean has been confirmed

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_TOKEN EJB_INACTIVE END_OF_BROWSE INVALID_BROWSE_TOKEN INVALID_POINTAT LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJBB gate, END_BROWSE Function

The END_BROWSE function of the EJBB gate ends the browse operation and deletes the browsetoken. This operation is available from EJJO and so the definitions must be consistent.

Input parameters

BROWSETOKEN The pointer which points to the first DJar in the chain to be deleted

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	EJB_INACTIVE INVALID_BROWSE_TOKEN LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR

EJBG gate, ADD_BEAN function

The ADD_BEAN function of the EJBG gate:

- Creates the Bean element in memory and chains it appropriately. The XRSINDI exit is called to notify the creation of this element.
- The ADDMODE parameter controls the compartmentalization of this operation for usage purposes (this defaults to NORMAL which does both the creation of the Control Block and its replacement in temporary mode). Usage of this verb is always via the Java layers and so ADDMODE(NORMAL) is used.
- The namespace for the Bean is at the CorbaServer level - the Bean cannot already have been installed from a different DJar (with the same name).
- The Bean itself is not actually installed until all the Beans within the Djar have been installed. Therefore, the Bean is added as a Control Block in a temporary state until all the Beans from the DJar have been so processed. When all the Beans in the DJar have been (successfully) added, the CONFIRM_ALL_BEANS call is made to alter the temporary state in the CB into a proper state. Otherwise the DELETE_ALL_BEANS call is made which deletes the Control Block. As the DJar was invalid it will be removed.
- On Warm restart, the owning Container and DJar will already have been restored from the Global Catalog.

Input parameters

BEAN Name of the Bean to be added

CORBASERVER Name of the CorbaServer for this Bean
DJAR Name of the Djar for this Bean
DDAREAFORIN Block for Bean deployment/meta data input
[ADDMODE] The type of create done for the Bean
[MESSAGE] Controls whether a message is issued when a CorbaServer is created

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BEAN_ALREADY_PRESENT CORBASERVER_ABSENT CORBASERVER_INVALID_STATE DDAREAFORIN_ABSENT DJAR_ABSENT DJAR_INVALID_STATE EJB_INACTIVE INVALID_BEAN INVALID_BROWSE_TOKEN INVALID_CORBASERVER INVALID_DDAREAFORIN INVALID_DD_ZERO_LENGTH INVALID_DD_ZERO_POINTER INVALID_DJAR LOCK_ERROR LOOP NAMESPACE_CONFLICT PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR

EJBG gate, ADD_BEAN_STATS function

The ADD_BEAN_STATS function of the EJBG gate increments the EJ domain's statistics counters for a specific enterprise bean.

Input parameters

BEAN Name of the enterprise bean whose statistics are to be incremented
CORBASERVER Name of the CorbaServer in which this bean is installed
[ACTIVATES] The number of times this bean has been activated
[PASSIVATES] The number of times this bean has been passivated
[CREATES] The number of times this bean has been created
[REMOVES] The number of times this bean has been removed
[METHOD_CALLS]
The number of method calls (other than the above) made against this bean

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

RESPONSE	Possible REASON values
EXCEPTION	BEAN_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJBG gate, CONFIRM_ALL_BEANS Function

The CONFIRM_ALL_BEANS function of the EJBG gate hardens all Beans associated with the given DJar within the relevant CorbaServer namespace. This just switches the state of a suitable Bean from temporary to normal. This will run when all Beans in the DJar have been correctly installed. The key is CS+DJar for this multiple status changing.

Input parameters

CORBASERVER The name of the CorbaServer for this Bean

DJAR The name of the DJar for this Bean

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BEAN_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJBG gate, DELETE_ALL_BEANS Function

The DELETE_ALL_BEANS function of the EJBG gate is executed when all of the Beans within the DJar did not install or when the owning DJar itself is deleted. All relevant Bean Control Blocks (whatever their state) are deleted. This works via the usual Browse mechanism (BROWSEMODE(ALL)) with POINTAT(PRIOR) enabled to delete each individual Bean. The key of CS+DJar+Bean is required.

Input parameters

CORBASERVER The name of the CorbaServer for this Bean

DJAR the name of the DJar for this Bean

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

RESPONSE	Possible REASON values
EXCEPTION	BEAN_ABSENT BROWSE_ERROR EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJBG gate, DELETE_BEAN Function

The DELETE_BEAN function of the EJBG gate deletes the Bean Control Block. The XRSINDI exit is also called to notify the removal. The full key of CS+DJar+Bean is required.

Input parameters

BEAN The name of the Bean
CORBASERVER The name of the CorbaServer for this Bean
DJAR The name of the DJar for the Bean

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BEAN_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR

EJBG gate, GET_BEAN_DD Function

The GET_BEAN_DD function of the EJBG gate returns the saved Deployment/Meta Data for the Bean (key is CS+Bean) in a buffer. The Buffer Pointer (_P) must be non-zero and point to a suitable area of length_M bytes. After the Deployment/Meta Data has been moved to the _P area, the _N item will be set to the length of the Deployment/Meta Data as is the DDLEN parameter. If _M is too small for the Deployment/Meta Data, then it is not moved into the _P area, _N is set to zero and the required length returned in DDLEN. Note that DDLEN and _N are only valid for OK operation or length errors - they are not available for use in other circumstances.

This operation is available via EJJO and so parameters should be kept consistent.

Input parameters

BEAN The name of the Bean
CORBASERVER The name of the CorbaServer for this Bean
DDAREAFORUPD A buffer for Bean the deployment/meta data update area

Output parameters

[DDLEN] The length of the deployment/meta data area. Used particularly to contain the length of the data if the size is larger than the maximum length for the ddareaforin block
[DJAR] The name of the DJar for this Bean
RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BEAN_ABSENT CORBASERVER_ABSENT CORBASERVER_INVALID_STATE DJAR_ABSENT DJAR_INVALID_STATE DDAREAFORUPD_ABSENT DD_AREA_TOO_SMALL EJB_INACTIVE INVALID_DDAREAFORUPD INVALID_DD_ZERO_LENGTH INVALID_DD_ZERO_POINTER LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJBG gate, INQUIRE_BEAN Function

The INQUIRE_BEAN function of the EJBG gate extracts information from the named Bean Control Block (key is CS+Bean). Note that the length of the Deployment/Meta Data is returned, but this XML is obtained via get_bean_dd.

This function can be used to determine the DJar which sourced the Bean.

Input parameters

BEAN The name for the Bean

CORBASERVER The name of the CorbaServer for this Bean

Output parameters

[DDLEN] The length of the deployment/meta data area. Used particularly to contain the length of the data if the size is larger than the maximum length of the ddareaforin block

[DJAR] The name of the DJarfor this Bean

[STATUS] The state of the Bean being inquired upon

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BEAN_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJBG gate, RESET_BEAN_STATS function

The RESET_BEAN_STATS function of the EJBG gate sets the EJ domain's statistics counters, for a specific enterprise bean, to zero.

Input parameters

BEAN Name of the enterprise bean whose statistics counters are to be reset

CORBASERVER Name of the CorbaServer in which this bean is installed

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BEAN_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJCB gate, START_BROWSE function

The START_BROWSE function of the EJCB gate initiates the browse upon the chain of CorbaServers. Positioning of the start of the Browse is not supported. Selection by CorbaServer is not provided. The end_browse condition is not returned if there are no suitable CorbaServers (this is postponed until the get_next). The returned browsetoken must be used for subsequent GET_NEXT operations.

Input parameters

None

Output parameters

BROWSETOKEN The pointer set up by START_BROWSE which points to the first CorbaServer in the chain to be browsed

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	EJB_INACTIVE LOCK_ERROR LOOP SETUP_ERROR STORAGE_ERROR

EJCB gate, GET_NEXT function

The GET_NEXT function of the EJCB gate returns the next CorbaServer Control Block in the list of CorbaServers. The ordering of CorbaServers returned is not specified (the order is not alpha order but Last-FirstOut for Browse purposes). The POINTAT parameter is used to enable a Browse to proceed when the aim of the Browse is to locate a CorbaServer to be deleted.

- POINTAT(NORMAL) should be used in all cases by the SPI layers and general users (and is the default).
- POINTAT(PRIOR) shows the deletion intent. POINTAT(PRIOR) should never be coded in normal circumstances and may result in an infinite loop if used without a delete.

Input parameters

BROWSETOKEN The pointer set up by START_BROWSE which points to the first CorbaServer in the chain to be browsed

[POINTAT] Indicates whether to advance the browse pointer to point to the next item in the chain

(NORMAL|PRIOR). NORMAL will return the next item in the chain, whereas PRIOR will always return the same item, unless that item has been deleted

Output parameters

CORBASERVER Name of the CorbaServer

[STATE] Indicates the current Resolution State and whether it is available for use or not.

[TIMEOUT] The elapsed time period (in seconds) of inactivity after which a session Bean can be discarded

[PORT] The binary TCP/IP port number for non-SSL communication included in IORs exported from this CorbaServer

[SSLPORT] The binary TCP/IP port number for SSL communication included in IORs exported from this CorbaServer

[SSL] the SSL type for this CorbaServer (YES|NO|CLIENTAUTH)

[JNDIPREFIX] The prefix to use at runtime when publishing the JNDI

[SHELF] The fully qualified name of a directory (a 'shelf' for 'jars') on HFS

[HOST] The TCP/IP hostname or the dotted decimal TCP/IP address included in IORs exported from this CorbaServer

[CERT] The label of a certificate within the keyring that is to be used as a client certificate in the SSL handshake for outbound IIOP connections

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BROWSE_TOKEN EJB_INACTIVE END_OF_BROWSE INVALID_BROWSE_TOKEN INVALID_POINTAT LOCK_ERROR LOOP SETUP_ERROR

EJCB gate, END_BROWSE function

The END_BROWSE function of the EJCB gate ends the browse operation and deletes the browsetoken.

Input parameters

BROWSETOKEN The pointer set up by START_BROWSE which points to the first CorbaServer in the chain to be browsed

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	EJB_INACTIVE INVALID_BROWSE_TOKEN LOCK_ERROR LOOP SETUP_ERROR STORAGE_ERROR

EJCG gate, ACTION_CORBASERVER function

The ACTION_CORBASERVER function of the EJCG gate is a gate which tells another party that something is to be done on the CorbaServer. The implemented actions are to manipulate the External Namespace for the named CorbaServer.

Input parameters

CORBASERVER Name of the CorbaServer on which the action is to be done

ACTIONMODE the action to perform on the CorbaServer. Possible values are:

`PUBLISH|RETRACT`

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	CORBASERVER_ABSENT CORBASERVER_INVALID_STATE DJAR_INVALID_STATE EJB_INACTIVE INVALID_ACTION LOCK_ERROR LOOP PARMS_STORAGE_ERROR PUBLISH_ERROR RETRACT_ERROR SETUP_ERROR STORAGE_ERROR

EJCG gate, ADD_CORBASERVER function

The ADD_CORBASERVER function creates a CorbaServer Control Block:

- creates the CorbaServer in memory, chains it appropriately, and saves an entry in the Global Catalog for Warm restart purposes.
- The XRSINDI exit is called to notify the creation of this element.
- The ADDMODE parameter controls the scope of this operation for restart purposes (this defaults to NORMAL which does both creation of the Control Block and its cataloging). Usage of this verb via the SPI/RDO layers should always code ADDMODE(NORMAL).

Input parameters

CORBASERVER Name of the CorbaServer to be added

STATE Indicates the current resolution state of the CorbaServer and whether it is available for use or not.

TIMEOUT The elapsed time (in seconds) of inactivity after which a session Bean can be discarded

JNDIPREFIX The prefix to use at runtime when publishing to JNDI

SHELF The fully qualified name of a directory (a 'shelf' for 'jars') on HFS

HOST The TCP/IP hostname or the dotted decimal TCP/IP address included in IORs exported from this CorbaServer

CERT The label of a certificate within the keyring that is to be used as a client certificate in the SSL handshake for outbound IIOP connections

[ADDMODE] The type of create done for the CorbaServer

[MESSAGE] Controls whether a message is issued when a CorbaServer is created

[ENABLESTATE]

Permissible values are:

`ENABLED|DISABLED|DISABLING`

[SCANINTERVAL]

[DJARDIR]
[AUTOPUBLISH]

[OUTPRIVACY]
[UNAUTH_TCPIPSERVICE]
The 8-character name of a TCPIPSERVICE that defines the characteristics of the port which is used for inbound IIOP with no authentication.

[SSLUNAUTH_TCPIPSERVICE]
The 8-character name of a TCPIPSERVICE that defines the characteristics of the port which is used for inbound IIOP with SSL but no client authentication.

[CLIENTCERT_TCPIPSERVICE]
The 8-character name of a TCPIPSERVICE that defines the characteristics of the port which is used for inbound IIOP with SSL client certificate authentication.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	ATTACH_ERROR CATALOG_ERROR CERT_ERROR CORBASERVER_ALREADY_THERE EJB_INACTIVE INVALID_CERT INVALID_CORBASERVER INVALID_HOST INVALID_JNDIPREFIX INVALID_PORT INVALID_SHELF INVALID_SSL INVALID_SSLPORT INVALID_STATE INVALID_TIMEOUT LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR

EJCG gate, AMEND_CORBASERVER function

The AMEND_CORBASERVER function of the EJCG gate changes information held within the CorbaServer Control Block. It does not harden this information over a CICS restart, nor does the change get communicated to the executing JVMs.

Input parameters

CORBASERVER
Name of the CorbaServer in which the information is to be changed

STATE
Indicates the current resolution state of the CorbaServer and whether it is available for use or not.

[TIMEOUT]
The elapsed time (in seconds) of inactivity after which a session Bean can be discarded

[STATE]
Specifies the state into which the CorbaServer is to be put. Values are

PENDINIT|INITING|UNUSABLE|PENDRESOLV|RESOLVING|INSERV|
UNRESOLVED|DELETING

[CURRENT_STATE]
Used as a check, must match the existing state of the CorbaServer. Values are

PENDINIT|INITING|UNUSABLE|PENDRESOLV|RESOLVING|INSERV|
UNRESOLVED|DELETING

[ENABLESTATE]
Permissible values are:

ENABLED|DISABLED|DISABLING

[SCANINTERVAL]

[DJARDIR]
[AUTOPUBLISH]

[OUTPRIVACY]
[UNAUTH_TCPIPSERVICE]
The 8-character name of a TCPIPSERVICE that defines the characteristics of the port which is used for inbound IIOP with no authentication.

[SSLUNAUTH_TCPIPSERVICE]
The 8-character name of a TCPIPSERVICE that defines the characteristics of the port which is used for inbound IIOP with SSL but no client authentication.

[CLIENTCERT_TCPIPSERVICE]
The 8-character name of a TCPIPSERVICE that defines the characteristics of the port which is used for inbound IIOP with SSL client certificate authentication.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	CORBASERVER_ABSENT CORBASERVER_INVALID_STATE CORBASERVER_STATE_CHANGED EJB_INACTIVE EJOS_ERROR INVALID_STATE INVALID_STATE_CHANGE INVALID_TIMEOUT LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJCG gate, DELETE_CORBASERVER function

The DELETE_CORBASERVER function of the EJCG gate removes a CorbaServer.

- Deletes the CorbaServer Control Block and removes the saved entry in the Global catalog. The XRSINDI exit is called to notify the removal.
- The Java layers are informed that the CorbaServer has been deleted.
- This operation has a side effect in that all DJars associated with the CorbaServer are also deleted, and then all the Beans from the DJar.

Input parameters

CORBASERVER Name of the CorbaServer to be deleted

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	CATALOG_ERROR CORBASERVER_ABSENT CORBASERVER_DELETING DELDJAR_ERROR EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR

EJCG gate, ESTABLISH function

The ESTABLISH function of the EJCG gate associates a CorbaServer with the calling task. It sets the task's Recovery Manager work token to reference the CorbaServer.

Input parameters

CORBASERVER Name of the CorbaServer with which an association is to be established

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	CORBASERVER_ABSENT CORBASERVER_DELETING EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR STORAGE_ERROR

EJCG gate, INQUIRE_CORBASERVER function

The INQUIRE_CORBASERVER function of the EJCG gate extracts information from the named CorbaServer Control Block. It is also executed indirectly from the EJJO gate.

Input parameters

CORBASERVER Name of the CorbaServer from which to extract information

Output parameters

[STATE] The state of the CorbaServer. Used to indicate whether the CorbaServer is available to use or not

[ENABLESTATE]

Possible values are:

ENABLED|DISABLED|DISABLING

[TIMEOUT] The elapsed time (in seconds) of inactivity after which a session Bean can be discarded

[PORT] The binary TCP/IP port number for non-SSL communication included in IORs exported from this CorbaServer

[SSLPORT] The binary TCP/IP port number for SSL communication included in IORs exported from this CorbaServer

[SSL] The SSL type for this CorbaServer (YES|NO|CLIENTAUTH)
[JNDIPREFIX] The prefix to use at runtime when publishing to JNDI
[SHELF] The fully qualified name of a directory (a 'shelf' for 'jars') on HFS
[HOST] The TCP/IP hostname or the dotted decimal TCP/IP address included in IORs exported from this CorbaServer
[CERT] The label of a certificate within the keyring that is to be used as a client certificate in the SSL handshake for outbound IIOP connections
RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	CORBASERVER_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJCG gate, RELINQUISH function

The RELINQUISH function of the EJCG gate ends an association between a CorbaServer and the calling task. It sets the task's Recovery Manager work token to blank.

Input parameters

CORBASERVER Name of the CorbaServer with which an association is to be ended
[ALLOC_COUNT]
The allocation number of the CorbaServer (used to prevent the accidental relinquishing of CorbaServers that have been freed and reallocated).

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	CORBASERVER_ABSENT CORBASERVER_DELETING EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR STORAGE_ERROR

EJCG gate, RESOLVE_CORBASERVER function

The RESOLVE_CORBASERVER function of the EJCG gate makes the CorbaServer available for use by Resolution (called by the CEJR transaction). The Java layers are informed that the CorbaServer has been created.

Input parameters

CORBASERVER Name of the CorbaServer

Output parameters

DID_STAGE The output from Resolve function which indicates which stage of resolution was done (STAGE1 or STAGE2)

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BAD_STATE_SET CATALOG_ERROR CORBASERVER_ABSENT CORBASERVER_INVALID_STATE EJB_INACTIVE EJOS_ERROR IILS_ERROR INVALID_CORBASERVER LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJCG gate, SET_ALL_STATE function

The SET_ALL_STATE function sets the state of all the CorbaServers.

input parameters

STATE Indicates the current Resolution State and whether it is available for use or not.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	EJB_INACTIVE INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJCG gate, WAIT_FOR_CORBASERVER function

The WAIT_FOR_CORBASERVER function of the EJCG gate will wait until the CorbaServer enters the required state.

Input parameters

CORBASERVER Name of the CorbaServer being waited on

STATE Indicates the current Resolution State and whether it is available for use or not.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	CORBASERVER_ABSENT CORBASERVER_UNRESOLVED CORBASERVER_UNUSABLE EJB_INACTIVE INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR WAIT_ERROR

EJCP gate, DISCARD_DJAR function

The DISCARD_DJAR function of the EJCP gate is called by the EJ domain during DJar Discard processing, but the DJar is unavailable for inquire.

The jar file is removed from its shelf and the shelf directory is deleted.

Input parameters

CORBASERVER Name of the CorbaServer (container) in which this DJar is installed. This is the CorbaServer name as specified in the DJar definition.

DJAR Name of the DJar to be discarded

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	SHELF_ACCESS_ERROR

EJCP gate, INSTALL_DJAR function

The INSTALL_DJAR function of the EJCP gate is called by the EJ domain when a DJar is installed as the second part of the DJar installation processing (after the DJar has been copied to the shelf).

The Java Container should initiate the processing of the DJar and pass the obtained Bean to the EJ domain.

The EJ domain expects to be called back for inquire_corbaserver and inquire_djar during the processing of this call, so the DJar must be available for inquire.

Input parameters

CORBASERVER Name of the CorbaServer (container) into which this DJar is to be installed. This is the CorbaServer name as specified in the DJar definition.

DJAR Name of the DJar to be installed

HFSFILE The fully qualified name of the jar file to be installed. The name must be a valid HFS filename and must not have any trailing blanks.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	CONTAINER_ERROR HFSFILE_NOT_FOUND HFS_ACCESS_ERROR SHELF_ACCESS_ERROR

EJCP gate, PRE_INSTALL_DJAR function

The PRE_INSTALL_DJAR function of the EJCP gate is called by the EJ domain when a DJar is installed to copy the hfsfile comprising the DJar onto the shelf.

The Java Container should create a copy of the DJar file on the shelf for the associated CorbaServer.

The EJ domain expects to be called back for inquire_corbaserver and inquire_djar during the processing of this call (EJJO gate), so the DJar must be available for inquire.

Input parameters

CORBASERVER Name of the CorbaServer (container) into which this DJar is to be installed. This is the CorbaServer name as specified in the DJar definition.

DJAR Name of the DJar to be installed

HFSFILE The fully qualified name of the jar file to be installed. The name must be a valid HFS filename and must not have any trailing blanks.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	CONTAINER_ERROR HFSFILE_NOT_FOUND HFS_ACCESS_ERROR SHELF_ACCESS_ERROR

EJCP gate, PUBLISH_CORBASERVER function

The PUBLISH_CORBASERVER function of the EJCP gate may be invoked by the EJ domain or the exec interface layer. All beans in the specified logical server are published to JNDI. This is not to be confused with the IILS publish function.

The Java container calls back to the EJ domain with a browse of all installed Beans for the logical server during processing of the function.

Input parameters

CORBASERVER Name of the CorbaServer for which to publish Beans.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

RESPONSE	Possible REASON values
EXCEPTION	CONTAINER_ERROR HFS_ACCESS_ERROR JNDI_ACCESS_ERROR

EJCP gate, PUBLISH_DJAR function

The PUBLISH_DJAR function of the EJCP gate may be invoked by the EJ domain or the exec interface layer. All beans in the specified logical server are published to JNDI.

The Java container calls back to the EJ domain with a browse of all installed Beans for the DJar during processing of the function.

Input parameters

CORBASERVER Name of the CorbaServer for which to publish Beans.

DJAR Name of the DJar for which to publish all Beans.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	CONTAINER_ERROR HFS_ACCESS_ERROR JNDI_ACCESS_ERROR

EJCP gate, RETRACT_CORBASERVER function

The RETRACT_CORBASERVER function of the EJCP gate may be invoked by the EJ domain or the exec interface layer. All beans in the specified logical server are retracted from JNDI. This is not to be confused with the IILS retract function.

The Java container calls back to the EJ domain with a browse of all installed Beans for the logical server during processing of the function.

Input parameters

CORBASERVER Name of the CorbaServer for which to retract Beans.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	HFS_ACCESS_ERROR JNDI_ACCESS_ERROR

EJCP gate, RETRACT_DJAR function

The RETRACT_DJAR function of the EJCP gate may be invoked by the EJ domain or the exec interface layer. All beans in the specified DJar server are retracted from JNDI.

The Java container calls back to the EJ domain with a browse of all installed Beans for DJar or the logical server during processing of the function.

Input parameters

CORBASERVER Name of the CorbaServer for which to retract Beans.

DJAR Name of the DJar for which to retract all Beans.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	HFS_ACCESS_ERROR JNDI_ACCESS_ERROR

EJDB gate, START_BROWSE function

The START_BROWSE function of the EJDB gate initiates the browse upon the chain of DJars. Positioning of the start of the Browse is not supported. Selection by DJars is not provided, but selection by owning CorbaServer is. The end_browse condition is not returned if there are no suitable DJars (this is postponed until the get_next). The returned browsetoken must be used for subsequent GET_NEXT operations.

Input parameters

CORBASERVER The name of the CorbaServer for this DJar

Output parameters

BROWSETOKEN The pointer set up by START_BROWSE which points to the first CorbaServer in the chain to be browsed

RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	EJB_INACTIVE INVALID_CORBASERVER LOCK_ERROR LOOP SETUP_ERROR STORAGE_ERROR

EJDB gate, GET_NEXT function

The GET_NEXT function of the EJDB gate returns the next DJar Control Block in the list of DJars that meets the selection criteria. The ordering of DJars returned is not specified (the order is not alpha order but LastIn-FirstOut for Browse purposes). The POINTAT parameter is used to enable a Browse to proceed when the aim of the browse is to locate a DJar to be deleted.

- POINTAT(NORMAL) should be used in all cases by the SPI layers and general users (and is the default).
- POINTAT(PRIOR) shows the deletion intent. POINTAT(PRIOR) should never be coded in normal circumstances and may result in an infinite loop if used without a delete.

Input parameters

BROWSETOKEN The pointer set up by START_BROWSE which points to the first DJar in the chain to be browsed

[POINTAT] Indicates whether to advance the browse pointer to point to the next item in the chain (NORMAL|PRIOR). NORMAL will return the next item in the chain, whereas PRIOR will always return the same item, unless that item has been deleted

Output parameters

DJAR The name of the DJar

[CORBASERVER]
The name of the CorbaSerever for this DJar

[HFSFILE] The fully qualified name of the deployed jar file on HFS

[STATE] The state of the DJar. Used to indicate whether the DJar is available for use or not

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BROWSE_TOKEN EJB_INACTIVE END_OF_BROWSE INVALID_BROWSE_TOKEN INVALID_POINTAT LOCK_ERROR LOOP SETUP_ERROR

EJDB gate, END_BROWSE function

The END_BROWSE function of the EJDB gate ends the browse operation and deletes the browsetoken.

Input parameters

BROWSETOKEN The pointer set up by START_BROWSE which points to the first DJar in the chain to be browsed

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	EJB_INACTIVE INVALID_BROWSE_TOKEN LOCK_ERROR LOOP SETUP_ERROR

EJDG gate, ACTION_DJAR function

The ACTION_DJAR function of the EJDG gate tells another party that something is to be done on the DJar. The implemented actions are to manipulate the External Namespace for the named DJar.

Input parameters

DJAR The name of the DJar

ACTIONMODE The action to perform on the DJar (PUBLISH|RETRACT)

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	DJAR_ABSENT DJAR_INVALID_STATE EJB_INACTIVE INVALID_ACTION LOCK_ERROR LOOP PARMS_STORAGE_ERROR PUBLISH_ERROR RETRACT_ERROR SETUP_ERROR STORAGE_ERROR

EJDG gate, ADD_DJAR function

The ADD_DJAR function of the EJDG gate creates a DJar Control Block.

- Creates the DJar element in memory, chains it appropriately, and saves an entry in the Global Catalog for Warm restart purposes.
- The XRSINDI exit is called to notify the creation of the element. The Java layers are informed that the DJar has been created.
- The ADDMODE parameter controls the compartmentalization of this operation for restart purposes (this defaults to NORMAL which does both the creation of the Control Block and its cataloging). Usage of this verb via the SPI/RDO layers should always code ADDMODE(NORMAL).

Input parameters

DJAR The name of the DJar to be added

CORBASERVER The name of the CorbaServer for this DJar

HFSFILE The fully qualified name of the deployed jar file on HFS

STATE Indicates the current Resolution State and whether it is available for use or not.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

RESPONSE	Possible REASON values
EXCEPTION	ATTACH_ERROR CATALOG_ERROR CORBASERVER_ABSENT CORBASERVER_INVALID_STATE DJAR_ALREADY_THERE EJB_INACTIVE INVALID_CORBASERVER INVALID_DJAR INVALID_HFSNAME INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR

EJDG gate, AMEND_DJAR function

The AMEND_DJAR function of the EJDG gate alters the DJar Control Block, but does not catalog the change or tell Java about the amendment.

Input parameters

DJAR The name of the DJar to be changed

[STATE] Specifies the state into which the DJar is to be put. Values are

`PENDINIT|INITING|UNUSABLE|PENDRESOLV|RESOLVING|INSERV|UNRESOLVED|DELETING`

[CURRENT_STATE]
Used as a check, and must match the existing state of the DJar. Values are

`PENDINIT|INITING|UNUSABLE|PENDRESOLV|RESOLVING|INSERV|UNRESOLVED|DELETING`

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BAD_STATE_CHANGE DJAR_ABSENT DJAR_STATE_CHANGED EJB_INACTIVE INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJDG gate, COUNT_FOR_CS function

The COUNT_FOR_CS function of the EJDG gate totals the number of DJars in each state for the owning CorbaServer

Input parameters

CORBASERVER The name of the CorbaServer

Output parameters

[NDJARS] The number of DJars in this Corbaserver

[NPENDINIT] The number of DJars which are in pendinit state in the CorbaServer

[NINITING] The number of DJars which are in initing state in the CorbaServer

[NUNUSABLE] The number of DJars which are in unusable state in the CorbaServer

[NPENDRESOLV]
The number of DJars which are in pendresolve state in the CorbaServer

[NRESOLVING] The number of DJars which are in resolving state in the CorbaServer

[NINSERV] The number of DJars which are in inservice state in the CorbaServer

[NUNRESOLVED]
The number of DJars which are in unresolved state in the CorbaServer

[NDELETING] The number of DJars which are in deleting state in the CorbaServer

RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BAD_STATE_CHANGE DJAR_ABSENT DJAR_STATE_CHANGED EJB_INACTIVE INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJDG gate, DELETE_DJAR function

The DELETE_DJAR function of the EJDG gate

- Deletes the DJar Control Block and removes the saved entry in the Global Catalog. the XRSINDI exit is also called to notify the removal.
- The Java layers are informed that the DJar has been deleted. However, this notification is not done if the deletion has been initiated by the deletion of the owning CorbaServer (this is notified by the delmode parameter - DELMODE(CASCADE) showing this CorbaServer initiated deletion and DELMODE(NORMAL) showing that the deletion has been initiated from the SPI/CEMT layers).
- This operation has a side effect in that all Beans associated with the DJar are also deleted.

Input parameters

DJAR The name of the DJar to be deleted

DELMODE Indicates what type of deletion is being done:
- DELMODE(CASCADE) indicates an owning CorbaServer initiated the deletion of this DJar
- DELMODE(NORMAL) indicated deletion is for SPI/CEMT delete DJar request

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

RESPONSE	Possible REASON values
EXCEPTION	CATALOG_ERROR DJAR_ABSENT DJAR_DELETING EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR

EJDG gate, DELETE_ALL_DJARS function

The DELETE_ALL_DJARS function of the EJDG gate is called when the owning CorbaServer is deleted which forces the cascaded deletion of all the DJars associated with the CorbaServer. This gate eventually uses EJDG.DELETE_DJAR with DELMODE(CASCADE) as part of its operation.

Input parameters

CORBASERVER The name of the CorbaServer for these DJars

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	CORBASERVER_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR

EJDG gate, INQUIRE_DJAR function

The INQUIRE_DJAR function of the EJDG gate extracts information from the named DJar Control Block

Input parameters

DJAR The name of the DJar

Output parameters

[CORBASERVER]
The name of the CorbaServer for this deployed jar file.

[HFSFILE] The fully qualified name of the deployed jar file on HFS

[STATE] The state of the deployed jar file. Used to indicate whether the deployed jar file is available for use or not

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

RESPONSE	Possible REASON values
EXCEPTION	DJAR_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJDG gate, SET_ALL_STATE function

The SET_ALL_STATE function of the EJDG gate sets the state of all the DJars.

Input parameters

STATE Indicates the current Resolution State and whether it is available for use or not.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	EJB_INACTIVE INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR

EJDG gate, WAIT_FOR_DJAR function

The WAIT_FOR_DJAR function of the EJDG gate waits until the DJars enter the required state.

Input parameters

DJAR The name of the DJar being waited on

STATE Indicates the current Resolution State and whether it is available for use or not.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	DJAR_ABSENT DJAR_UNRESOLVED DJAR_UNUSABLE EJB_INACTIVE INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR WAIT_ERROR

EJDG gate, WAIT_FOR_USABLE_DJARS function

The WAIT_FOR_USABLE_DJARS function of the EJDG gate waits until all the DJars associated with a CorbaServer are INSERV.

Input parameters

CORBASERVER The name of the CorbaServers for these DJars

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	CORBASERVER_ABSENT CORBASERVER_ERROR CORBASERVER_INVALID_STATE COUNT_ERROR DJAR_ABSENT DJAR_UNRESOLVED DJAR_UNUSABLE EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR WAIT_ERROR

EJDI gate, ADD_ENTRY Function

The ADD_ENTRY function of the EJDI gate adds a new entry to the Directory partition for the specified LogicalServer.

No entry with the same name should exist in the specified LogicalServer partition. In the case of a transaction entry, no existing entry should refer to the same request stream, but this is not checked.

Input parameters

ENTRY_KEY The key (OTS or Object Key) for the entry

ENTRY_TYPE Indicates whether this is a transaction or object_key entry

LOGICALSERVER
Name of the LogicalServer for which the entry is to be added

REQUEST_STREAM_ID
Public ID of the request stream to be put in the entry

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE_ENTRY FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_FULL_ERROR FILE_IO_ERROR STORE_NOT_OPEN
INVALID	INVALID_KEYLENGTH

EJDI gate, INITIALIZE Function

The INITIALIZE function of the EJDI gate is called when a store_not_open has been detected.

Input parameters

LOGICALSERVER
Name of the LogicalServer for which the entry is to be added

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	CICS_TERMINATING FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_NOT_FOUND FILE_RECOVERY_ERROR FILE_RECOVERY_UNKNOWN CTL_REC_FULL_ERROR
INVALID	INVALID_KEYLENGTH INVALID_RECORD_SIZE

EJDI gate, LOOKUP_ENTRY Function

The LOOKUP_ENTRY function of the EJDI gate looks up the given OTS transaction or object key / LogicalServer pair and returns the associated Request Stream if found.

Input parameters

ENTRY_KEY The key (OTS transaction or Object Key) for the entry to be returned

ENTRY_TYPE Indicates whether this is a transaction or object_key entry

LOGICALSERVER
Name of the LogicalServer to search for the entry

REQUEST_STREAM_BUFFER
Caller supplied buffer to contain the request stream id

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

RESPONSE	Possible REASON values
EXCEPTION	BUFFER_TOO_SMALL ENTRY_NOT_FOUND FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR OBJECT_CORRUPT STORE_NOT_FOUND STORE_NOT_OPEN
INVALID	INVALID_KEYLENGTH

EJDI gate, REMOVE_ENTRY Function

The REMOVE_ENTRY function of the EJDI gate removes a transaction or object key for a given LogicalServer.

Input parameters

ENTRY_KEY The key (OTS transaction or Object Key) for the entry to be removed

ENTRY_TYPE Indicates whether this is a transaction or object_key entry

LOGICALSERVER
Name of the LogicalServer for which the entry is to be removed

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	ENTRY_NOT_FOUND FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR STORE_NOT_OPEN
INVALID	INVALID_KEYLENGTH

EJDU gate, DUMP_DATA Function

The DUMP_DATA function of the EJDU gate is used to collect data from a dumping class. It will be placed in the chain of data collected by EJDU and formatted out when a CICS dump occurs.

Input parameters

DATA A pointer and length pair containing the data to be stored for inclusion in a dump.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INTERNAL_ERROR
EXCEPTION	INSUFFICIENT_STORAGE
INVALID	INVALID_FORMAT INVALID_FUNCTION

EJDU gate, DUMP_STACK Function

The DUMP_STACK function of the EJDU gate is used to collect the stack of a running JVM. The stack is passed as a string to EJDU and will be formatted out separately from the other data collected by EJDU's DUMP_DATA function. This function should be called before DUMP_DATA as it will free any existing data gathered for the running task.

Input parameters

DATA
A pointer and length pair containing the data to be stored for inclusion in a dump.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INTERNAL_ERROR
EXCEPTION	INSUFFICIENT_STORAGE
INVALID	INVALID_FORMAT INVALID_FUNCTION

EJDU gate, INQUIRE_TRACE_FLAGS Function

The INQUIRE_TRACE_FLAGS function of the EJDU gate is used to return the current settings of all the trace flags. It takes into account the master trace flag setting when returning the result. The trace flags are returned as a continuous block of storage with 2 bytes for each flag, in domain order.

Input parameters

TRACE_DATA
A block of data containing the trace flags in domain order, where each trace flag takes up 2 bytes

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INTERNAL_ERROR
EXCEPTION	BAD_DOMAIN_TOKEN TRACE_BUFFER_ERROR
INVALID	INVALID_FORMAT INVALID_FUNCTION

EJGE gate, INITIALIZE Function

The INITIALIZE function of the EJGE gate creates the various things in the EJE Anchor Block (Locks, Store Subpools, Statii etc.) and then sets up the initial chains of CorbaServer, DJar and BEan Control Blocks (and the Browse equivalents). These chains all start with a dummy X'00' element and end with another dummy X'FF' element. This permits easy chaining and detection of end-of-lists. However, more importantly, this technique enables multi-TCB operations to proceed as there are never any EJ Element wide-locks - all locks are at the CorbaServer, DJar or Bean level. After the EJE anchor block has been setup it is never subsequently amended.

Input parameters

STARTTYPE The startup type for this CICS system (WARM|COLD)

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	CATALOG_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR

EJGE gate, QUIESCE Function

The QUIESCE function of the EJGE gate runs when a CEMT P SHUT is executed.

Input parameters

None

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR

EJGE gate, TERMINATE Function

The TERMINATE function of the EJGE gate runs when a CEMT P IMMED is executed.

Input parameters

None

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR

EJIO gate, RESOLVE Function

The RESOLVE function of the EJIO gate controls the operation of Resolution processing. It is called by the CEJR transaction.

Input parameters

None

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BEAN_ADD_ERROR CATALOG_ERROR EJB_INACTIVE ENV_ERROR LOOP MULTIUSE OBJECTSTORE_ERROR PARMS_STORAGE_ERROR PRIORFAIL RESC_BAD_STB RESC_GETNEXT_ERROR RESD_BAD_STB RESD_GETNEXT_ERROR RESOLV_FAIL_CS RESOLV_FAIL_DJAR SETUP_ERROR STORAGE_ERROR

EJIO gate, RESOLVE_CSERVERS Function

The RESOLVE_CSERVERS function of the EJIO gate scans all existing CorbaServer Control Blocks that have not been fully processed and issues a EJCG.RESOLVE_CORBASERVER on the first such CorbaServer. (both Stage one 'copying the DJar to the Shelf' and Stage two 'Opening Object Stores' Resolution Processing).

Input parameters

None

Output parameters

EJIO gate, RESOLVE_DJARS Function

The RESOLVE_DJARS function of the EJIO gate scans all existing DJar Control Blocks that have not been fully processed and issues a EJDG.RESOLVE_DJAR on the first such DJar (both Stage one 'copying the DJar to the Shelf' and Stage two 'Bean loading' Resolution Processing).

Input parameters

None

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BEAN_ADD_ERROR CATALOG_ERROR EJB_INACTIVE ENV_ERROR LOCK_ERROR LOOP OBJECTSTORE_ERROR PARMS_STORAGE_ERROR RESD_BAD_STB RESD_GETNEXT_ERROR SETUP_ERROR STORAGE_ERROR

EJJO gate

The EJJO Gate provides a subset of the aforementioned EJ Domain Gates for direct use by the CICS-supplied layers residing within Java code. See the native functions for details.

EJMI gate, ADD_BEAN Function

The ADD_BEAN function of the EJMI gate adds the named Bean within the named CorbaServer to the EJMI state.

A duplicate_bean exception is returned if there is already a Bean of that name within the given CorbaServer. (The DJar must be discarded before the Bean can be added again.)

Input parameters

BEAN The name of the Bean to be added
CORBASERVER The name of the CorbaServer containing the Bean
DJAR The name of the DJar

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE_BEAN

EJMI gate, ADD_METHOD Function

The ADD_METHOD function of the EJMI gate adds the information for the named method within the given Bean and CorbaServer.

An unknown_bean exception is returned if there the given Bean and CorbaServer combination is not present in the EJMI state.

A duplicate_method exception is returned if there is already a method of that name within the given Bean and CorbaServer combination.

Input parameters

BEAN The name of the Bean to be added
CORBASERVER The name of the CorbaServer containing the Bean
METHOD The name of the method

XCOORD Indicates whether an external OTS transaction coordinator, if there is one, is respected for determining transaction commit or rollback. Values are:

RESPECTED|IGNORED

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE_METHOD UNKNOWN_BEAN

EJMI gate, DISCARD_METHOD_INFO Function

The DISCARD_METHOD_INFO function of the EJMI gate removes from the given CorbaServer all the information about Beans with the given DJar name. If no DJar name is specified all Beans are removed.

Input parameters

CORBASERVER The name of the CorbaServer for this DJar

[DJAR] The name of the DJar for these Beans

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_CORBASERVER

EJMI gate, GET_METHOD_INFO Function

The GET_METHOD_INFO function of the EJMI gate returns the information about the named method within the named Bean and CorbaServer.

An unknown_method exception is returned if the method is not found within the Bean and CorbaServer combination.

Input parameters

BEAN The name of the Bean

CORBASERVER The name of the CorbaServer for this Bean

METHOD The name of the Bean method

Output parameters

XCOORD Indicates whether an external OTS transaction coordinator, if there is one, is respected for determining transaction commit or rollback Values are:

RESPECTED|IGNORED

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_BEAN UNKNOWN_CORBASERVER UNKNOWN_METHOD

EJMI gate, INITIALIZE Function

The INITIALIZE function of the EJMI gate initializes the EJMI state in the EJ anchor block.

Input parameters

None

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

EJOB gate, END_BROWSE_OBJECT Function

The END_BROWSE_OBJECT function of the EJOB gate is called after START_BROWSE_OBJECT to end the Browse of a file or object_store.

Input parameters

BROWSE_TOKEN The token returned by START_BROWSE

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
INVALID	INVALID_TOKEN

EJOB gate, GET_NEXT_OBJECT Function

The GET_NEXT_OBJECT function of the EJOB gate is called after START_BROWSE_OBJECT to return the next object in the file or object_store.

Input parameters

BROWSE_TOKEN The token returned by START_BROWSE

[KEY_BUFFER] A buffer in which the next object key is returned

[OBJECT_BUFFER]
A buffer in which the next object is returned

Output parameters

[ACTIVE_TIMEOUT]
A full-word giving the number of seconds after which Objects in the Active state may be automatically deleted from the store.

[FILE_NAME] The 8-character name of the file containing the Object Store.

[LAST_UPDATED]
The time in STCK seconds when the object was last stored or activated.

[OBJECT_SIZE]
The size of the object being inquired.

[PASSIVE_TIMEOUT]
A full-word giving the number of seconds after which Objects in the Passive state may be automatically deleted from the store.

[STATUS] The state of the Object, ACTIVE or PASSIVE.

[STORE_NAME] The 8-character name of the Object Store.

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BUFFER_TOO_SMALL END_BROWSE FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND OBJECT_CORRUPT
INVALID	INVALID_TOKEN

EJOB gate, INQUIRE_OBJECT Function

The INQUIRE_OBJECT function of the EJOB gate is called to return the Object data and attributes associated with the given key.

Input parameters

STORE_NAME The 8-character name of the Object Store

KEY_BLOCK A block giving the key of the Object being inquired

[OBJECT_BUFFER]
A buffer in which the object is returned

Output parameters

[ACTIVE_TIMEOUT]
A full-word giving the number of seconds after which Objects in the Active state may be automatically deleted from the store.

[FILE_NAME] The 8-character name of the file containing the Object Store.

[LAST_UPDATED]
The time in STCK seconds when the object was last stored or activated.

[OBJECT_SIZE]
The size of the object being inquired.

[PASSIVE_TIMEOUT]
A full-word giving the number of seconds after which Objects in the Passive state may be automatically deleted from the store.

[STATUS] The state of the Object, ACTIVE or PASSIVE.

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BUFFER_TOO_SMALL FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND OBJECT_CORRUPT OBJECT_NOT_FOUND STORE_NOT_OPEN

RESPONSE	Possible REASON values
INVALID	INVALID_KEYLENGTH

EJOB gate, INQUIRE_STORES Function

The INQUIRE_STORES function of the EJOB gate is called to return a list of the Object Store names associated with the given file. The list is returned as an array of 8-character store names.

Input parameters

[FILE_NAME] The optional 8-character name of the file to be inquired. If omitted then the default file 'DFHEJOS' will be used.

[OBJECT_BUFFER]
A buffer in which the array of store names is returned.

[SUBPOOL] A storage subpool from which to getmain the object block.

Output parameters

STORE_COUNT The number of store names being returned

[OBJECT_BLOCK]
A block containing the array of 8-character store names. If specified then SUBPOOL must also be specified.

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BUFFER_TOO_SMALL FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND FILE_REC_SIZE_ERROR
INVALID	INVALID_TOKEN

EJOB gate, RETRIEVE_STATISTICS Function

The RETRIEVE_STATISTICS function of the EJOB gate is called by statistics to return the statistics associated with a supplied store key.

Input parameters

STORE_NAME The name of the store for which the statistics are being retrieved.

[DATA] A flag indicating if the statistics must be returned

[OBJECT_BUFFER]
The statistics buffer which the data must be put into

[RESET] A flag indicating that the statistics fields must be reset

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

RESPONSE	Possible REASON values
EXCEPTION	BUFFER_TOO_SMALL BUFFER_NOT_SUPPLIED STORE_NOT_OPEN

EJOB gate, START_BROWSE_OBJECT Function

The START_BROWSE_OBJECT function of the EJOB gate is called To Browse an Object Store. If STORE_NAME is omitted then all Objects in the file are browsed. If FILE_NAME is omitted then the default file 'DFHEJOS' is assumed.

Input parameters

[STORE_NAME] The 8-character name of the Object Store to browse
[FILE_NAME] The 8-character name of the Object Store to browse

Output parameters

BROWSE_TOKEN A token required by GET_NEXT and END_BROWSE
RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|PURGED
[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND FILE_REC_SIZE_ERROR STORE_NOT_FOUND

EJOS gate, ACTIVATE_OBJECT Function

The ACTIVATE_OBJECT function of the EJOS gate is called to Activate an Object instance.

If DELETE(NO) is specified then the ACTIVATE function will mark Objects as ACTIVE in the store. ACTIVE_TIMEOUT indicates when entries marked ACTIVE may be automatically deleted from the store. If DELETE(YES) is specified then ACTIVE_TIMEOUT is ignored, and the ACTIVATE function will delete the object from the store.

Input parameters

STORE_NAME The 8-character name of the Object Store
KEY_BLOCK A block giving the key of the Object to be activated
OBJECT_BUFFER
The buffer into which the Object is returned
DELETE YES means the Object is to be deleted from the while and NO means the Object is to be marked ACTIVE in the file

Output parameters

OBJECT_SIZE The size of the Object being activated
RESPONSE is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|PURGED
[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	BUFFER_TOO_SMALL FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR OBJECT_CORRUPT OBJECT_IS_ACTIVE OBJECT_NOT_FOUND STORE_NOT_OPEN
INVALID	INVALID_KEYLENGTH

EJOS gate, CLOSE_OBJECT_STORE Function

The CLOSE_OBJECT_STORE function of the EJOS gate is called to Close an Object Store in the local system.

If an Object Store is open with a non-zero timeout value, then a task is scheduled to sweep the store periodically, deleting timed-out Objects. It will, therefore, improve CICS performance if stores are closed when not required.

Input parameters

STORE_NAME
The 8-character name of the Object Store

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values

`OK|EXCEPTION|DISASTER|INVALID|PURGED`

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	STORE_NOT_OPEN

EJOS gate, OPEN_OBJECT_STORE Function

The OPEN_OBJECT_STORE function of the EJOS gate is called to Open a new or existing Object Store in the local system.

An Object Store must be opened in each region wishing to use it. Many object stores can use the same CICS file, or they can each specify a different file.

If an Object Store of the same name is already open in that region, the existing definition is replaced, and the new file name and timeout values are then used. As timeout values are stored with the object, changes to the store definition will not affect objects already stored. If file_name is omitted, then the default file 'DFHEJOS' will be used.

Input parameters

ACTIVATE_TIMEOUT
A full-word giving the number of seconds after which Objects in the Active State may be automatically deleted from the store

PASSIVE_TIMEOUT
A full-word giving the number of seconds after which Objects in the Passive State may be automatically deleted from the store

RECOVERY
YES indicates that the file should be recoverable. If it is not, FILE_RECOVERY_ERROR is returned. NO indicates that the file should not be recoverable. If it is then

FILE_RECOVERY_ERROR is returned. If CICS is unable to determine whether the file is recoverable then FILE_RECOVERY_UNKNOWN is returned

STORE_NAME The 8-character name of the Object Store

[FILE_NAME] The optional 8-character name of the file to be used

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	CICS_TERMINATING FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND FILE_RECOVERY_ERROR FILE_RECOVERY_UNKNOWN FILE_REC_SIZE_ERROR CTL_REC_FULL_ERROR
INVALID	INVALID_OBJECT_TIMEOUT

EJOS gate, REMOVE_OBJECT Function

The REMOVE_OBJECT function of the EJOS gate is called to Remove an Object instance from the specified Object Store.

Input parameters

KEY_BLOCK A block containing the key of the Object to be removed

STORE_NAME The 8-character name of the Object Store

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND OBJECT_NOT_FOUND STORE_NOT_OPEN
INVALID	INVALID_KEYLENGTH

EJOS gate, REMOVE_STORE Function

The REMOVE_STORE function of the EJOS gate is called to Remove one or all Object Stores from the specified file.

When a Store is removed, it should be removed or closed in every region in which it is open. If not, then data may be lost.

Input parameters

STORE_NAME |ALL
: The 8-character name of the Object Store

[FILE_NAME]
: The optional 8-character name of the file to be used

Output parameters

RESPONSE
: is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON]
: is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND STORE_NOT_OPEN

EJOS gate, STORE_OBJECT Function

The STORE_OBJECT function of the EJOS gate is called to Store an Object instance.

The Object is identified by a KEY of from 1 to (recordsize -64) bytes, and the Object can be of any size.

If no Object with that key exists in the store then one is created in the Passive state. If an Object with the same key already exists in the Store, then the action depends on the REPLACE value (YES|NO|ACTIVE|PASSIVE). An exception OBJECT_IS_ACTIVE or OBJECT_IS_PASSIVE indicates why an object was not replaced.

Input parameters

STORE_NAME
: The 8-character name of the Object Store

REPLACE
: Yes means that an Object with the same key will be replaced. NO means that an Object with the same key will not be replaced. ACTIVE means that an ACTIVE Object with the same key is replaced. PASSIVE means that a PASSIVE Object with the same key is replaced. Values are

YES|NO|ACTIVE|PASSIVE

KEY_BLOCK
: A block containing the key of the Object to be stored

OBJECT_BLOCK
: A block containing the Object data to be stored

Output parameters

RESPONSE
: is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON]
: is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_FULL_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND OBJECT_IS_ACTIVE OBJECT_IS_PASSIVE STORE_NOT_OPEN

RESPONSE	Possible REASON values
INVALID	INVALID_KEYLENGTH

EJSO gate, INQUIRE_CORBASERVER function

The INQUIRE_CORBASERVER function of the EJSO gate is used by the EJ domain to find any TCPIP parameters that are also kept in the corba server after resolution time. This function is used by JAVA code and normal CICS code.

Input parameters

CORBASERVER
The 4 character name of the corba server.

Output parameters

ASSERTED_PORT
A fullword containing the port number of the TCPIPSERVICE named in the ASSERTED attribute of the CORBASERVER.

ASSERTED_PRIVACY
An enumerated type of REQUIRED|SUPPORTED|NOTSUPPORTED taken from the TCPIPSERVICE named in the ASSERTED attribute of the CORBASERVER.

ASSERTED_SSL
An enumerated type of YES|CLIENTAUTH taken from the TCPIPSERVICE named in the ASSERTED attribute of the CORBASERVER.

ASSERTED_HASH
A fullword created by the sockets domain to represent the TCPIPSERVICE named in the ASSERTED attribute of the CORBASERVER.. It is used to check that the TCPIPSERVICE in the listener region has the same attributes as the one in the AOR.

BASIC_PORT
A fullword containing the port number of the TCPIPSERVICE named in the BASIC attribute of the CORBASERVER.

BASIC_PRIVACY
An enumerated type of REQUIRED|SUPPORTED|NOTSUPPORTED taken from the TCPIPSERVICE named in the BASIC attribute of the CORBASERVER.

BASIC_SSL
An enumerated type of YES|CLIENTAUTH taken from the TCPIPSERVICE named in the BASIC attribute of the CORBASERVER.

BASIC_HASH
A fullword created by the sockets domain to represent the TCPIPSERVICE named in the BASIC attribute of the CORBASERVER. It is used to check that the TCPIPSERVICE in the listener region has the same attributes as the one in the AOR.

UNAUTH_PORT
A fullword containing the port number of the TCPIPSERVICE named in the UNAUTH attribute of the CORBASERVER.

UNAUTH_PRIVACY
An enumerated type of REQUIRED|SUPPORTED|NOTSUPPORTED taken from the TCPIPSERVICE named in the UNAUTH attribute of the CORBASERVER.

UNAUTH_SSL
An enumerated type of YES|NO|CLIENTAUTH taken from the TCPIPSERVICE named in the UNAUTH attribute of the CORBASERVER.

UNAUTH_HASH
A fullword created by the sockets domain to represent the TCPIPSERVICE named in the UNAUTH attribute of the CORBASERVER. It is used to check that the TCPIPSERVICE in the listener region has the same attributes as the one in the AOR.

CLIENTCERT_PORT
A fullword containing the port number of the TCPIPSERVICE named in the CLIENTCERT attribute of the CORBASERVER.

CLIENTCERT_PRIVACY
An enumerated type of REQUIRED|SUPPORTED|NOTSUPPORTED taken from the TCPIPSERVICE named in the CLIENTCERT attribute of the CORBASERVER.

CLIENTCERT_SSL
An enumerated type of YES|CLIENTAUTH taken from the TCPIPSERVICE named in the CLIENTCERT attribute of the CORBASERVER.

CLIENTCERT_HASH
A fullword created by the sockets domain to represent the TCPIPSERVICE named in the

CLIENTCERT attribute of the CORBASERVER. It is used to check that the TCPIPSERVICE in the listener region has the same attributes as the one in the AOR.

SSLUNAUTH_PORT
A fullword containing the port number of the TCPIPSERVICE named in the SSLUNAUTH attribute of the CORBASERVER.

SSLUNAUTH_PRIVACY
An enumerated type of REQUIRED|SUPPORTED|NOTSUPPORTED taken from the TCPIPSERVICE named in the SSLUNAUTH attribute of the CORBASERVER.

SSLUNAUTH_SSL
An enumerated type of clientauth taken from the TCPIPSERVICE named in the SSLUNAUTH attribute of the CORBASERVER.

SSLUNAUTH_HASH
A fullword created by the sockets domain to represent the TCPIPSERVICE named in the SSLUNAUTH attribute of the CORBASERVER. It is used to check that the TCPIPSERVICE in the listener region has the same attributes as the one in the AOR.

RESPONSE
is the domain's response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CORBASERVER_ABSENT LOCK_ERROR
DISASTER	ABEND

EJSO gate, AMEND_CORBASERVER function

The AMEND_CORBASERVER function of the EJSO gate is used by the EJ domain to update TCPIP parameters that are also kept in the corba server after resolution time. This function is only used by DFHEJCG RESOLVE_CORBASERVER.

Input parameters

CORBASERVER
The 4 character name of the corba server.

ASSERTED_PORT
A fullword containing the port number of the TCPIPSERVICE named in the ASSERTED attribute of the CORBASERVER.

ASSERTED_PRIVACY
An enumerated type of REQUIRED|SUPPORTED|NOTSUPPORTED taken from the TCPIPSERVICE named in the ASSERTED attribute of the CORBASERVER.

ASSERTED_SSL
An enumerated type of YES|CLIENTAUTH taken from the TCPIPSERVICE named in the ASSERTED attribute of the CORBASERVER.

ASSERTED_HASH
A fullword created by the sockets domain to represent the TCPIPSERVICE named in the ASSERTED attribute of the CORBASERVER.. It is used to check that the TCPIPSERVICE in the listener region has the same attributes as the one in the AOR.

BASIC_PORT
A fullword containing the port number of the TCPIPSERVICE named in the BASIC attribute of the CORBASERVER.

BASIC_PRIVACY
An enumerated type of REQUIRED|SUPPORTED|NOTSUPPORTED taken from the TCPIPSERVICE named in the BASIC attribute of the CORBASERVER.

BASIC_SSL
An enumerated type of YES|CLIENTAUTH taken from the TCPIPSERVICE named in the BASIC attribute of the CORBASERVER.

BASIC_HASH
A fullword created by the sockets domain to represent the TCPIPSERVICE named in the BASIC attribute of the CORBASERVER. It is used to check that the TCPIPSERVICE in the listener region has the same attributes as the one in the AOR.

UNAUTH_PORT
A fullword containing the port number of the TCPIPSERVICE named in the UNAUTH attribute of the CORBASERVER.

UNAUTH_PRIVACY
An enumerated type of REQUIRED|SUPPORTED|NOTSUPPORTED taken from the TCPIPSERVICE named in the UNAUTH attribute of the CORBASERVER.

UNAUTH_SSL
An enumerated type of YES|NO|CLIENTAUTH taken from the TCPIPSERVICE named in the UNAUTH attribute of the CORBASERVER.

UNAUTH_HASH
A fullword created by the sockets domain to represent the TCPIPSERVICE named in the UNAUTH attribute of the CORBASERVER. It is used to check that the TCPIPSERVICE in the listener region has the same attributes as the one in the AOR.

CLIENTCERT_PORT
A fullword containing the port number of the TCPIPSERVICE named in the CLIENTCERT attribute of the CORBASERVER.

CLIENTCERT_PRIVACY
An enumerated type of REQUIRED|SUPPORTED|NOTSUPPORTED taken from the TCPIPSERVICE named in the CLIENTCERT attribute of the CORBASERVER.

CLIENTCERT_SSL
An enumerated type of YES|CLIENTAUTH taken from the TCPIPSERVICE named in the CLIENTCERT attribute of the CORBASERVER.

CLIENTCERT_HASH
A fullword created by the sockets domain to represent the TCPIPSERVICE named in the CLIENTCERT attribute of the CORBASERVER. It is used to check that the TCPIPSERVICE in the listener region has the same attributes as the one in the AOR.

SSLUNAUTH_PORT
A fullword containing the port number of the TCPIPSERVICE named in the SSLUNAUTH attribute of the CORBASERVER.

SSLUNAUTH_PRIVACY
An enumerated type of REQUIRED|SUPPORTED|NOTSUPPORTED taken from the TCPIPSERVICE named in the SSLUNAUTH attribute of the CORBASERVER.

SSLUNAUTH_SSL
An enumerated type of clientauth taken from the TCPIPSERVICE named in the SSLUNAUTH attribute of the CORBASERVER.

SSLUNAUTH_HASH
A fullword created by the sockets domain to represent the TCPIPSERVICE named in the SSLUNAUTH attribute of the CORBASERVER. It is used to check that the TCPIPSERVICE in the listener region has the same attributes as the one in the AOR.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CORBASERVER_ABSENT LOCK_ERROR
DISASTER	ABEND

The Enterprise Java (EJ) domain is logically divided into three parts:

- Elements, which covers the manipulation of the EJ Resources of CorbaServers (EJCG), DJars (EJDG) and Beans (EJBJ)
- Object Stores, used to store stateful Session Beans, and to hold the EJB Directory (EJOS and EJOB)
- Directory, used to record the association of OTS transactions and object instances with Request Processors (EJDI).

EJ domain's generic gates

Table 52 summarizes the EJ domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 52. EJ domain's generic gates

Gate	Trace	Function	Format
EJDM	EJ 01xx	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
EJST	EJ 04xx	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

For descriptions of these functions and their input and output parameters, refer to the Sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—Chapter 78, "Domain manager domain (DM)," on page 663

Format STST—"System programming command flows" on page 264

The Domain Manager gates perform normal internal state initialization and termination functions.

Modules

Module	Function
DFHEJBB	Bean Browse EJBB Gate
DFHEJBG	Bean General EJBG Gate
DFHEJCB	CorbaServer Browse EJCB Gate
DFHEJCG	CorbaServer General EJCG Gate
DFHEJCP	Command Processor functions EJCP Gate
DFHEJDB	DJar Browse EJDB Gate
DFHEJDG	DJar General EJDG Gate
DFHEJDI	EJB Directory EJDI Gate
DFHEJDM	EJ Initialize/Terminate EJDM Gate
DFHEJDU	EJ Dump Interface EJDU Gate
DFHEJGE	EJ General Initialization/Termination functions EJGE Gate
DFHEJIO	CEJR Resolution EJIO Gate
DFHEJJO	Jave Interface EJJO Gate
DFHEJMI	Method Information function EJMI Gate
DFHEJOB	Object Store Browse EJOB Gate
DFHEJOS	Object Store General EJOS Gate
DFHEJST	Statistics General EJSt Gate

Exits

NONE

Trace

The point IDs have just been added onto the usual EJ ones.

Chapter 83. Event manager domain (EM)

The event manager domain manages event and timer objects created within CICS BTS activities. For further information regarding these objects see *CICS Business Transaction Services*.

Event manager domain's specific gates

Table 53 summarizes the event manager domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 53. Event manager domain's specific gates

Gate	Trace	Function	XPI
EMEM	EM 0201	ADD_SUBEVENT	NO
	EM 0202	CHECK_TIMER	NO
		DEFINE_ATOMIC_EVENT	NO
		DEFINE_COMPOSITE_EVENT	NO
		DEFINE_TIMER	NO
			NO
		DELETE_EVENT	NO
		DELETE_TIMER	NO
		FIRE_EVENT	NO
		FORCE_TIMER	NO
		INQUIRE_STATUS	NO
		REMOVE_SUBEVENT	NO
		RESET_EVENT	NO
		RETRIEVE_REATTACH_EVENT	NO
		RETRIEVE_SUBEVENT	NO
		TEST_EVENT	
EMBR	EM 0301	INQUIRE_EVENT	NO
	EM 0302	START_BROWSE_EVENT	NO
		GET_NEXT_EVENT	NO
		END_BROWSE_EVENT	NO
		INQUIRE_TIMER	NO
		START_BROWSE_TIMER	NO
		GET_NEXT_TIMER	NO
		END_BROWSE_TIMER	NO

EMEM gate, ADD_SUBEVENT function

The ADD_SUBEVENT function adds a subevent to an existing composite event.

Input parameters

EVENT is the name of the composite event.

SUBEVENT is the name of the subevent.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY EVENT_NOT_FOUND INVALID_EVENT_TYPE SUBEVENT_NOT_FOUND INVALID_SUBEVENT

EMEM gate, CHECK_TIMER function

The CHECK_TIMER function returns the status of a timer.

Input parameters

TIMER_NAME is the name of the timer.

Output parameters

TIMER_STATUS returns the status of the timer. It can have one of these values:

EXPIRED|FORCED|UNEXPIRED

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY TIMER_NOT_FOUND

EMEM gate, DEFINE_ATOMIC_EVENT function

The DEFINE_ATOMIC_EVENT function defines an atomic event of type ACTIVITY or INPUT.

Input parameters

EVENT is the name of the event.

EVENT_TYPE is the type of the event. It can have one of these values:

ACTIVITY|INPUT

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY INVALID_EVENT_NAME DUPLICATE_EVENT

EMEM gate, DEFINE_COMPOSITE_EVENT function

The DEFINE_COMPOSITE_EVENT function defines a composite event with an associated predicate which may be AND or OR. Up to eight subevents may be provided.

Input parameters

EVENT is the name of the composite event.

PREDICATE is the predicate type. It may have either one of these values:

AND|OR

SUBEVENT_LIST
is an optional list of up to 8 subevents.

Output parameters

SUBEVENT_IN_ERROR
returns the number of the first subevent which is in error (if any).

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY INVALID_EVENT_NAME SUBEVENT_NOT_FOUND INVALID_SUBEVENT DUPLICATE_EVENT

EMEM gate, DEFINE_TIMER function

The DEFINE_TIMER function defines a timer.

Input parameters

TIMER_NAME is the name of the timer.
EVENT is the optional name of an event to be associated with the timer.
AFTER indicates whether or not the timer is an interval. It may have either of these values:
YES|NO
AT indicates whether or not the timer is a time. It may have either of these values:
YES|NO
DAYS is the number of days for an interval.
HOURS is the number of hours for an interval or time.
MINUTES is the number of minutes for an interval or time.
SECONDS is the number of seconds for an interval or time.
ON indicates whether or not a date has been specified. It may have either of these values:
YES|NO
YEAR is the year.
MONTH is the month.
DAYOFMONTH is the day of the month.
DAYOFYEAR is the day of the year.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED
[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY INVALID_TIMER_NAME DUPLICATE_TIMER INVALID_EVENT_NAME DUPLICATE_EVENT INVALID_INTERVAL INVALID_TIME

EMEM gate, DELETE_EVENT function

The DELETE_EVENT function deletes an event.

Input parameters

EVENT is the name of the event to be deleted.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED
[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY EVENT_NOT_FOUND INVALID_EVENT_TYPE

EMEM gate, DELETE_TIMER function

The DELETE_TIMER function deletes a timer.

Input parameters

TIMER is the name of the timer to be deleted.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY TIMER_NOT_FOUND

EMEM gate, FIRE_EVENT function

The FIRE_EVENT function causes an event to fire.

Input parameters

EVENT is the name of the event to be fired.

EVENT_VERSION
is an optional version number for the event.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY EVENT_NOT_FOUND INVALID_EVENT_TYPE ALREADY_FIRED VERSION_NOT_FOUND

EMEM gate, FORCE_TIMER function

The FORCE_TIMER function causes a timer to expire early.

Input parameters

TIMER is the name of the timer to be forced.

ACQUIRED_PROCESS
indicates whether or not the timer to be forced is owned by the acquired process. It may have either of these values:

YES|NO

ACQUIRED_ACTIVITY
indicates whether or not the timer to be forced is owned by the acquired activity. It may have either of these values:

YES|NO

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY NO_ACQUIRED_PROCESS NO_ACQUIRED_ACTIVITY INVALID_ACTIVITY TIMER_NOT_FOUND

EMEM gate, INQUIRE_STATUS function

The INQUIRE_STATUS function returns the status of the event pool for the current activity.

Output parameters

PENDING_EVENTS
indicates whether any events are pending. It may have either of these values:

YES|NO

PENDING_ACTIVITY_EVENTS
indicates whether any activity events are pending. It may have either of these values:

YES|NO

REATTACH
indicates whether the task should be reattached. It may have either of these values:

YES|NO

EVENTS_PROCESSED
indicates whether any events were processed during this activation. It may have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY

EMEM gate, REMOVE_SUBEVENT function

The REMOVE_SUBEVENT function removes a subevent from the named composite event.

Input parameters

EVENT
is the name of the composite event.

SUBEVENT
is the name of the subevent.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY EVENT_NOT_FOUND INVALID_EVENT_TYPE SUBEVENT_NOT_FOUND INVALID_SUBEVENT

EMEM gate, RETRIEVE_REATTACH_EVENT function

The RETRIEVE_REATTACH_EVENT function retrieves the next event from the current activity's reattach queue.

Output parameters

EVENT is the name of the retrieved reattach event.

EVENT_TYPE is the type of the retrieved reattach event. It may have one of the following values:

ACTIVITY|COMPOSITE|INPUT|SYSTEM|TIMER

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY END_EVENTS

EMEM gate, RETRIEVE_SUBEVENT function

The RETRIEVE_SUBEVENT function retrieves the next event from the named composite event's subevent queue.

Input parameters

EVENT is the name of the composite event.

Output parameters

SUBEVENT is the name of the subevent.

EVENT_TYPE is the type of the retrieved reattach event. It may have one of the following values:

ACTIVITY|COMPOSITE|INPUT|SYSTEM|TIMER

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY EVENT_NOT_FOUND INVALID_EVENT_TYPE END_SUBEVENTS NO_SUBEVENTS

EMEM gate, TEST_EVENT function

The TEST_EVENT function returns the fire status of the named event.

Input parameters

EVENT is the name of the event to be tested.

Output parameters

FIRED returns the fire status of the event. It may have either of these values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_CURRENT_ACTIVITY EVENT_NOT_FOUND

EMBR gate, INQUIRE_EVENT function

The INQUIRE_EVENT function returns information about the named event.

Input parameters

EVENT is the name of the event being inquired upon.

Output parameters

EVENT_TYPE is the type of the event. It can have one of these values:

ACTIVITY|COMPOSITE|INPUT|SYSTEM|TIMER

FIRED returns the fire status of the event. It may have either of these values:

YES|NO

PREDICATE is the predicate type (for composite events only). It may have either one of these values:

AND|OR

PARENT is the name of the parent (if the event is a subevent).

TIMER_NAME is the name of the associated timer (if the event is of type timer).

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_ACTIVITY_ID NO_CURRENT_ACTIVITY FILE_NOT_AUTH EVENT_NOT_FOUND READ_FAILURE FILE_UNAVAILABLE

EMBR gate, START_BROWSE_EVENT function

The START_BROWSE_EVENT function starts an event browse and returns a token to be used for the browse.

Input parameters

ACTIVITY_ID is an optional activity id for the activity whose event pool is to be browsed.

Output parameters

BROWSE_TOKEN returns a token which is used to identify the browse.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_ACTIVITY_ID FILE_NOT_AUTH NO_CURRENT_ACTIVITY READ_FAILURE FILE_UNAVAILABLE

EMBR gate, GET_NEXT_EVENT function

The GET_NEXT_EVENT function returns the next name in the browse specified by the browse token, and returns the attributes associated with the event.

Input parameters

BROWSE_TOKEN is a token which identifies the browse.

Output parameters

EVENT is the name of the event.

EVENT_TYPE is the type of the event. It can have one of these values:

ACTIVITY|COMPOSITE|INPUT|SYSTEM|TIMER

FIRED returns the fire status of the event. It may have either of these values:

YES|NO

PREDICATE is the predicate type (for composite events only). It may have either one of these values:

AND|OR

PARENT is the name of the parent (if the event is a subevent).

TIMER_NAME is the name of the associated timer (if the event is of type timer).

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN BROWSE_END

EMBR gate, END_BROWSE_EVENT function

The END_BROWSE_EVENT function ends the event browse identified by the browse token.

Input parameters

BROWSE_TOKEN is a token which identifies the browse.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN

EMBR gate, INQUIRE_TIMER function

The INQUIRE_TIMER function returns information about the named timer.

Input parameters

TIMER is the name of the timer being inquired upon.

Output parameters

EVENT is the name of the associated event.

TIMER_STATUS is the status of the timer. It can have one of these values:

EXPIRED|FORCED|UNEXPIRED

ABSTIME returns the timer's expiry time in ABSTIME format.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_ACTIVITY_ID NO_CURRENT_ACTIVITY FILE_NOT_AUTH TIMER_NOT_FOUND READ_FAILURE FILE_UNAVAILABLE

EMBR gate, START_BROWSE_TIMER function

The START_BROWSE_TIMER function starts a timer browse and returns a token to be used for the browse.

Input parameters

ACTIVITY_ID is an optional activity id for the activity whose event pool is to be browsed.

Output parameters

BROWSE_TOKEN returns a token which is used to identify the browse.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_ACTIVITY_ID FILE_NOT_AUTH NO_CURRENT_ACTIVITY READ_FAILURE FILE_UNAVAILABLE

EMBR gate, GET_NEXT_TIMER function

The GET_NEXT_TIMER function returns the next name in the browse specified by the browse token, and returns the attributes associated with the timer.

Input parameters

BROWSE_TOKEN is the token which identifies the browse.

Output parameters

TIMER is the name of the timer.

EVENT is the name of the associated event.

TIMER_STATUS is the status of the timer. It can have one of these values:

EXPIRED|FORCED|UNEXPIRED

ABSTIME returns the timer's expiry time in ABSTIME format.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN BROWSE_END

EMBR gate, END_BROWSE_TIMER function

The END_BROWSE_TIMER function ends the timer browse identified by the browse token.

Input parameters

BROWSE_TOKEN is a token which identifies the browse.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN

Event manager domain's generic gates

Table 54 summarizes the event manager domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 54. Event manager domain's generic gates

Gate	Trace	Function	Format
DMDM	EM 0101 EM 0102	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
EMBA	EM 0401 EM 0402	INQUIRE_DATA_LENGTH GET_DATA DESTROY_TOKEN	BAGD

For descriptions of these functions and their input and output parameters, refer to the Sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669

In initialization, quiesce, and termination processing, the event manager domain performs only internal routines.

Modules

Module	Function
DFHEMDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN

Module	Function
DFHEMEM	Handles the following requests: ADD_SUBEVENT CHECK_TIMER DEFINE_ATOMIC_EVENT DEFINE_COMPOSITE_EVENT DEFINE_TIMER DELETE_EVENT DELETE_TIMER FIRE_EVENT FORCE_TIMER INQUIRE_STATUS REMOVE_SUBEVENT RESET_EVENT RETRIEVE_REATTACH_EVENT RETRIEVE_SUBEVENT TEST_EVENT
DFHEMBR	Handles the following requests: INQUIRE_EVENT START_BROWSE_EVENT GET_NEXT_EVENT END_BROWSE_EVENT INQUIRE_TIMER START_BROWSE_TIMER GET_NEXT_TIMER END_BROWSE_TIMER
DFHEMBA	Handles the following requests: INQUIRE_DATA_LENGTH GET_DATA DESTROY_TOKEN
DFHEMDUF	Formats the EM domain control blocks
DFHEMTRI	Interprets EM domain trace entries

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the event manager domain are of the form EM xxxx; the corresponding trace levels are EM 1, EM 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 84. IP ECI (IE) domain

The IP ECI (IE) domain processes external call interface (ECI) requests that arrive from a CICS client that is connected to CICS by a TCP/IP network. It attaches a mirror task to issue the appropriate program link request, and returns the results to the client.

IE domain's generic gates

Table 55 summarizes the IE domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 55. IE domain's generic gates

Gate	Trace	Function	FORMAT
DMDM	IE 0100 IE 0101	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM DMDM DMDM

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669

IE domain's specific gates

Table 56 summarizes the IE domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 56. IE domain's specific gates

Gate	Trace	Function	XPI
IEIE	IE 0001 IE 0002	PROCESS_ECI_FLOW RECEIVE SEND SEND_ERROR	NO NO NO NO

IEIE gate, PROCESS_ECI_FLOW function

Initiates processing of a flow from an ECI client, either by attaching a new mirror task, or by posting an existing mirror task.

Input parameters

None

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_INSTALLED INVALID_FLOW INSTALL_FAILED
DISASTER	ABEND FREEMAIN_FAILURE RECEIVE_FAILURE SEND_FAILURE
INVALID	INVALID_FORMAT INVALID_FUNCTION

IEIE gate, RECEIVE function

Receives input from an ECI client.

Input parameters

None

Output parameters

DATA_ADDRESS
The address of the buffer containing the data received.

DATA_LENGTH
The length of the data received.

CODEPAGE
The codepage of the request

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CLIENT_NOT_RESPONDING REQUEST_PURGED
DISASTER	ABEND FREEMAIN_FAILURE INVALID_REQUEST WAIT_FAILURE
INVALID	INVALID_FORMAT INVALID_FUNCTION

IEIE gate, SEND function

Sends a reply to an ECI client.

Input parameters

DATA_ADDRESS
The address of the buffer containing the data to be sent. DATA_LENGTH.

DATA_LENGTH
The length of the data to be sent.

LAST
This is the last send in this conversation, or not.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ABEND REQUEST_PURGED
DISASTER	FREEMAIN_FAILURE INVALID_REQUEST SEND_FAILURE
INVALID	INVALID_FORMAT INVALID_FUNCTION

IEIE gate, SEND_ERROR function

Sends an FMH7 to an ECI client.

Input parameters

MESSAGE_NUMBER
The number of the IE component message to be sent to the client.

INSERT1
The first message insert

INSERT2
The second message insert

INSERT3
The third message insert

INSERT4
The fourth message insert

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND FREEMAIN_FAILURE INVALID_REQUEST SEND_FAILURE
INVALID	INVALID_FORMAT INVALID_FUNCTION

Control blocks

Each installed client is represented by an IP ECI Client State Block (IECSB). This is the IP equivalent of the TCSE extension built by CICS for SNA connected clients.

The IECSBs are indexed by the socket token provided by SO domain.

Each client can have many simultaneous conversations in progress. Each of these is represented by an IP ECI Client Conversation Block (IECCB). The IECCBs are indexed by a session id allocated by the client and the sequence number of the attach that initiated the conversation.

Modules

DFHIEIE
: The main part of IE domain. Processes all DFHIEIE_GATE functions.

DFHIEDM
: IE domain initialization and termination.

Exits

None

Trace

The point ids for the IP ECI domain are of the form IE xxxx. The corresponding trace levels are IE 1, IE 2 and Exc. IE level 2 tracing includes a complete trace of all data sent to and from the client.

Chapter 85. IIOP domain (II)

The IIOP domain represents the non-Java portion of the IIOP EJB support encompassing the following:

- Request Receiver
- Request Handler
- Request Processor
- Request Models
- Command Processor

IIOP domain's specific gates

Table 57 summarizes the IIOP domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 57. IIOP domain's specific gates

Gate	Trace	Function	XPI
IIRR	II 0100 II 0101	PROCESS_REQUESTS	NO
IIRH	II 0200 II 0201	FIND_REQUEST_STREAM PARSE	NO NO
IIMM	II 0300 II 0301	ADD_REPLACE_RQMODEL DELETE_RQMODEL COMMIT_RQMODELS	NO NO NO
IIRQ	II 0400 II 0401	INQUIRE_RQMODEL START_BROWSE GET_NEXT END_BROWSE MATCH_RQMODEL	NO NO NO NO NO
IIRP	II 0700 II 0701	GET_INITIAL_DATA RECEIVE_REQUEST INVOKE SEND_REPLY RECEIVE_REPLY INITIALISE TERMINATE	NO NO NO NO NO NO NO
IIRS	II 0900 II 0901	HANDLE_SECURITY_CONTEXT DESTROY_VAULT	NO NO
IICP	II 0800 II 0801	ABSTRACT	NO

IILS gate, ADD_LOGICAL_SERVER function

The ADD_LOGICAL_SERVER function of the IILS gate is used to invoke JAVA code to add a logical server.

Input parameters

logical_server
: Name of the logical server to be added.

shelf
: Name of the shelf to which the logical server is to be added.

Output parameters

RESPONSE
: is the domain's response to the call. It can have any of these values:

  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
: is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SHELF_ACCESS_ERROR

IILS gate, DELETE_LOGICAL_SERVER function

The DELETE_LOGICAL_SERVER function of the IILS gate is used to invoke JAVA code to delete a logical server.

Input parameters

logical_server
Name of the logical server to be deleted.

shelf
Name of the shelf to which the logical server is to be deleted.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND SHELF_ACCESS_ERROR

IILS gate, PUBLISH_LOGICAL_SERVER function

The PUBLISH_LOGICAL_SERVER function of the IILS gate is used to invoke JAVA code to publish a logical server to JNDI.

Input parameters

logical_server
Name of the logical server to be published.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND JNDI_ERROR

IILS gate, RETRACT_LOGICAL_SERVER function

The RETRACT_LOGICAL_SERVER function of the IILS gate is used to invoke JAVA code to retract a logical server from the JNDI.

Input parameters

logical_server
Name of the logical server to be retracted.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND JNDI_ERROR

IIMM gate, ADD_REPLACE_RQMODEL function

The ADD_REPLACE_RQMODEL function of the IIMM gate is used to install or delete and install a request model.

Input parameters

rqmodel_name Name of the request model to be installed.

corbaserver Name of the corbaserver for this request model.

model_type can be any of the following values: EJB | CORBA | GENERIC

[module_pattern]
If CORBA - name of the module.

[interface_pattern]
If CORBA - name of the interface.

[bean_pattern]
If EJB - name of the bean.

[interface_type]
If EJB - HOME | REMOTE | BOTH

operation_pattern
name of the operation.

tranid Transaction name.

catalog YES | NO. If YES the request model is added to the catalogue.

Output parameters

duplicate_model_name
Name of a model with the same parameters if DUPLICATE_PATTERN set.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_NAME DUPLICATE_PATTERN

IIMM gate, DELETE_RQMODEL function

The DELETE_RQMODEL function of the IIMM gate is used to delete an installed request model.

Input parameters

rqmodel_name Name of the request model to be deleted.

Output parameters

RESPONSE is the domain's response to the call it can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND

IIMM gate, COMMIT_RQMODELS function

The COMMIT_RQMODELS function of the IIMM gate is used to commit the request model to the catalogue.

input parameters

commit_token Token for catalogue writes.

Output parameters

RESPONSE
is the domain's response to the call it can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND

IIRH gate, FIND_REQUEST_STREAM function

The FIND_REQUEST_STREAM function of the IIRH gate is used to examine the incoming GIOP request and to find a new or existing request stream using request models and the directory.

It also handles any incoming security contexts.

Input parameters

request_block
Address and length of the GIOP Request - the block must contain the whole of the request header. It need not contain the body.

[urm_commarea_block]
Storage used as input to the security user-replaceable program.

[urmname]
Name of the security user-replaceable program.

[userid]
userid to be used by the ORB.

[force_create]
YES | NO - YES indicates that IIRH must CREATE a new request stream. NO indicates that normal logic is used to see if a request stream exists and to JOIN it if it does or CREATE a new one if it does not.

[vault_ptr_addr]
The address of the vault pointer, updated in DFHIIRS.

[authentication_type]
An enumerated type set from the AUTHENTICATION attribute of the TCPIPSERVICE definition. Values are

NONE|BASIC|ASSERTED|CERTIFICATE

Output parameters

request_stream_token
The token, representing the request stream, to be used as input for the SEND_REQUEST.

result
JOINED | CREATED - whether the request stream was joined or created.

RESPONSE
is the domain's response to the call it can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	URM_DENIED_PERMISSION URM_USERID_NOTAUTH INVALID_OBJECT_KEY REQUEST_ERROR

IIRH gate, PARSE function

The PARSE function of the IIRH gate is used to examine the incoming GIOP request or reply and to return selected information in the output parameters.

Input parameters

request_block
: Address and length of the GIOP Request/reply - the block must contain the whole of the request/reply header. It need not contain the body.

Output parameters

[response_expected]
: YES | NO - YES is set if the response_expected bit is on in the request header.

[reply_status]
: NO_EXCEPTION | USER_EXCEPTION | SYSTEM_EXCEPTION | LOCATION_FORWARD This extracts the reply status from a reply header.

[codeset_context]
: This is a block containing a pointer to and the length of the named context if it exists within the request or reply. The pointer and length are set to 0 if the context does not exist.

[sending_context]
: This is a block containing a pointer to and the length of the named context if it exists within the request or reply. The pointer and length are set to 0 if the context does not exist.

[tracking_context]
: This is a block containing a pointer to and the length of the named context if it exists within the request or reply. The pointer and length are set to 0 if the context does not exist.

[connection_context]
: This is a block containing a pointer to and the length of the named context if it exists within the request or reply. The pointer and length are set to 0 if the context does not exist.

[redirection_context]
: This is a block containing a pointer to and the length of the named context if it exists within the request or reply. The pointer and length are set to 0 if the context does not exist.

[requestId] is the requestId extracted from the request or reply header.

RESPONSE is the domain's response to the call it can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	REQUEST_ERROR

IIRP gate, GET_INITIAL_DATA function

The GET_INITIAL_DATA function of the IIRP gate is used by the ORB program DFJIIRP (or its CICS-key equivalent DFJIIRQ) to set up an environment to allow it to issue further IIRP requests and to return the output parameters below.

Input parameters

none

Output parameters

rp_token token to allow further calls for the same Request Processor

server_name name of the corba server held by the request stream for the incoming request.

public_id The public_id that identifies the request stream for the incoming request.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ERROR_REENTERED ABEND
EXCEPTION	REQUEST_STREAM_NOT_CURRENT NO_SERVER_DATA NO_PUBLIC_ID

IIRP gate, RECEIVE_REQUEST function

The RECEIVE_REQUEST function of the IIRP gate is used by the ORB program DFJIIRP (or its CICS-key equivalent DFJIIRQ) to receive a request via a request stream from a Request Receiver. This is for INBOUND requests.

Input parameters

RP_TOKEN Token supplied by GET_INITIAL_DATA or INITIALISE representing state storage.

CONTINUE YES | NO - YES is set if RECEIVE_REQUEST is to listen for a further request.

REQUEST_BUF A buffer, into which the received request is to be placed.

RECEIVE_TYPE FULL | OVERFLOW - FULL is set for the first receive_request. OVERFLOW is set if the buffer supplied to the first receive_request was too small.

Output parameters

correlation_id
The correlation id returned by the request stream receive_request.

bytes_available
Set if BUFFER_TOO_SMALL is set. It contains the actual size of of the buffer needed which is obtained from the GIOP header received by receive_request.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ERROR_REENTERED ABEND
EXCEPTION	RECEIVE_REQUEST_FAILED LISTEN_FAILED MESSAGE_NOT_RECEIVABLE BUFFER_TOO_SMALL TIMEOUT_NOTIFIED GIOP_REQ_HEADER_INVALID REQUEST_INVALID
INVALID	INVALID_RP_TOKEN

IIRP gate, INVOKE function

The INVOKE function of the IIRP gate is used by the ORB program DFJIIRP (or its CICS-key equivalent DFJIIRQ) to send an outbound request and to receive its reply.

Input parameters

RP_TOKEN Token supplied by GET_INITIAL_DATA or INITIALISE representing state storage.

RS_TOKEN Token representing the outbound request stream.

CONTINUE YES | NO - YES is set if RECEIVE_REQUEST is to listen for a further request.

REQUEST_BUF A block holding the request to be sent.

REPLY_BUF A buffer, into which the reply is to be placed.

Output parameters

bytes_available
Set if BUFFER_TOO_SMALL is set. It contains the actual size of of the buffer needed for the reply which is obtained from the GIOP reply header received by INVOKE

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ERROR_REENTERED ABEND
EXCEPTION	REQUEST_RECEIVED LISTEN_FAILED MESSAGE_NOT_RECEIVABLE RECEIVE_REPLY_FAILED SEND_REQUEST_FAILED BUFFER_TOO_SMALL TIMEOUT_NOTIFIED GIOP_REP_HEADER_INVALID GIOP_FRAGMENT_NOT_EXPECTED GIOP_CLOSE_CONN_RECEIVED GIOP_FRAGMENT_EXPECTED GIOP_INVALID_VERSION GIOP_INVALID_MESSAGE_TYPE GIOP_MESSAGE_ERROR_RCVD REQUEST_INVALID REDIRECTION_RECEIVED
INVALID	INVALID_RP_TOKEN

IIRP gate, RECEIVE_REPLY function

The RECEIVE_REPLY function of the IIRP gate is used by the ORB program DFJIIRP (or its CICS-key equivalent DFJIIRQ) to receive an outbound reply to an outbound request. It is used, following INVOKE, if INVOKE indicated that a further request was ready before the reply was available (loopback) or if the reply buffer supplied by INVOKE was too small.

Input parameters

RP_TOKEN
Token supplied by GET_INITIAL_DATA or INITIALISE representing state storage.

RS_TOKEN
Token representing the outbound request stream.

RECEIVE_TYPE
FULL | OVERFLOW. FULL is set if all of the reply is to be received. OVERFLOW is set if the previous INVOKE or RECEIVE_REPLY ended in BUFFER_TOO_SMALL.

REPLY_BUF
A buffer, into which the reply is to be placed.

Output parameters

bytes_available
Set if BUFFER_TOO_SMALL is set. It contains the actual size of of the buffer needed for the reply which is obtained from the GIOP reply header received by INVOKE.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ERROR_REENTERED ABEND

RESPONSE	Possible REASON values
EXCEPTION	REQUEST_RECEIVED LISTEN_FAILED MESSAGE_NOT_RECEIVABLE RECEIVE_REPLY_FAILED SEND_REQUEST_FAILED BUFFER_TOO_SMALL TIMEOUT_NOTIFIED GIOP_REP_HEADER_INVALID GIOP_FRAGMENT_NOT_EXPECTED GIOP_CLOSE_CONN_RECEIVED GIOP_FRAGMENT_EXPECTED GIOP_INVALID_VERSION GIOP_INVALID_MESSAGE_TYPE GIOP_MESSAGE_ERROR_RCVD REQUEST_INVALID REDIRECTION_RECEIVED
INVALID	INVALID_RP_TOKEN

IIRP gate, SEND_REPLY function

The SEND_REPLY function of the IIRP gate is used by the ORB program DFJIIRP (or its CICS-key equivalent DFJIIRQ) to send a reply via a request stream to an inbound request.

Input parameters

RP_TOKEN Token supplied by GET_INITIAL_DATA or INITIALISE representing state storage.

REPLY_BUF A buffer containing the reply to be sent.

CORRELATION_ID
of the request returned by IIRP RECEIVE_REQUEST.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	SEND_REPLY_FAILED
INVALID	INVALID_RP_TOKEN

IIRP gate, INITIALISE function

The INITIALISE function of the IIRP gate is used by the ORB program DFJIIRP (or its CICS-key equivalent DFJIIRQ) to set up an environment to allow it to issue further IIRP requests. This is used during COMMAND PROCESSING. For example when DFJIIRQ is processing an ADD_CORBASERVER command.

Input parameters

rp_token token to allow further calls for the same Request Processor.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ERROR_REENTERED ABEND

IIRP gate, TERMINATE function

The TERMINATE function of the IIRP gate is used by the ORB program DFJIIRP (or its CICS-key equivalent DFJIIRQ) in normal and command processing mode to free any storage obtained by GET_INITIAL_DATA or INITIALISE. If necessary, it will also leave the request stream.

Input parameters

rp_token token to allow further calls for the same Request Processor. If rp_token is 0 then TERMINATE checks to see if this task has any state storage to be freed.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ERROR_REENTERED ABEND

IIRQ gate, INQUIRE_RQMODEL function

The INQUIRE_RQMODEL function of the IIRQ gate is used to inquire on a particular model, returning the output parameters below.

Input parameters

rqmodel_name Name of the request model for which information is needed.

Output parameters

[corbaserver]
Name of the corbaserver for this request model.

[model_type] EJB | CORBA | GENERIC

[module_pattern]
If CORBA - name of the module.

[interface_pattern]
If CORBA - name of the interface.

[bean_pattern]
If EJB - name of the bean.

[interface_type]
If EJB - HOME | REMOTE | BOTH

[operation_pattern]
name of the operation.

[tranid] Transaction name.

RESPONSE is the domain's response to the call it can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	NOT_FOUND

IIRQ gate, START_BROWSE function

The START_BROWSE function of the IIMM gate is used to return a token to allow all the request models to be browsed.

Input parameters

none

Output parameters

browse_token token to be used by get_next and end_browse.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NONE

IIRQ gate, GET_NEXT function

The GET_NEXT function of the IIMM gate is used to pass back the output parameters for the next request model.

Input parameters

browse_token token created by start_browse representing the next request model

Output parameters

rqmodel_name Name of the request model to be installed.

[corbaserver]
Name of the corbaserver for this request model.

[model_type] EJB | CORBA | GENERIC

[module_pattern]
If CORBA - name of the module.

[interface_pattern]
If CORBA - name of the interface.

[bean_pattern]
If EJB - name of the bean.

[interface_type]
If EJB - HOME | REMOTE | BOTH

[operation_pattern]
name of the operation.

[tranid] Transaction name.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END

IIRQ gate, END_BROWSE function

The END_BROWSE function of the IIMM gate is used to end the browse session.

Input parameters

browse_token token created by start_browse.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	none

IIRQ gate, MATCH_RQMODEL function

The MATCH_RQMODEL function of the IIRQ gate is used to find the most specific request model that matches the input parameters.

Input parameters

corbaserver Name of the corbaserver for this request model.

[module_name_block]
If CORBA - name of the module.

[interface_name_block]
If CORBA - name of the interface.

[bean_name_block]
If EJB - name of the bean.

[interface_type]
If EJB - HOME | REMOTE | BOTH

operation_pattern
name of the operation.

Output parameters

[tranid] Transaction name from the matching model. If no match is found the default of CIRP is returned.

RESPONSE is the domain's response to the call it can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	none

IIRR gate, PROCESS_REQUESTS function

The PROCESS_REQUESTS function of the IIRR gate is used to receive a GIOP request from a socket, find a request stream, send the request over the request stream, optionally receive a reply and send the reply to the socket. This process continues until the socket is closed or no further data is available.

Input parameters

none

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ERROR_REENTERED ABEND

RESPONSE	Possible REASON values
EXCEPTION	RESCHEDULE NO_PERMISSION GIOP_FRAGMENT_NOT_EXPECTED GIOP_CLOSE_CONN_RECEIVED GIOP_FRAGMENT_EXPECTED GIOP_INVALID_HEADER GIOP_INVALID_VERSION GIOP_INVALID_MESSAGE_TYPE GIOP_MESSAGE_ERROR_RCVD GIOP_REPLY_RECEIVED IIRH_FIND_EXCEPTION SOCK_RECEIVE_TIMEOUT

IIRS gate, HANDLE_SECURITY_CONTEXT function

The HANDLE_SECURITY_CONTEXT function of the IIRS gate is used by the Request Handler to check that any relevant security context in an incoming request is the correct one one and to validate the parameters therein

This function is handed a security context which it verifies (depending on the msgType) and extracts the userid and password etc..If an error is found it sets msgType and ErrorCode into the security context and returns information to allow DFHIIRR to build a system exception.

Input parameters

SECURITY_CONTEXT
A buffer containing the address and length of the security context to be checked.

If an exception response is returned the security context will have been updated to be an error security context.

AUTHENTICATION_TYPE
An enumerated type containing the TCPIPSERVICE AUTHENTICATION value - in other words, what sort of security context is expected. This may be

BASIC|ASSERTED

VAULT_PTR_ADDR
The address of the start of the vault chain. The vault contains sessionID to userid mappings and is added to, looked up in if the security context is BASIC.

CORBASERVER
ASSERTED IDENTITY needs the corbaserver name as input to a security check.

Output parameters

USERID
The main aim of this module is to return a USERID for the ORB (DFJIIRP) task to run under. Only returned with an OK response.

STRING
If an exception response is returned, STRING contains an enumerated type to be used in the STRING section of the system exception written to the client by DFHIIRR: for example, if the STRING returned is NO_PERMISSION, then the string NO_PERMISSION is added to the system_exception reply. Values are

NO_PERMISSION|INTERNAL|MARSHAL

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
AUTHENTICATION_TYPEis returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	SECURITY_CHECK_FAILED

IIRS gate, DESTROY_VAULT function

The DESTROY_VAULT function of the IIRS gate is used by the Request Receiver to delete any security vault entries that have built up for this connection.

Input parameters

VAULT_PTR_ADDR
: The address of the start of the vault entry chain to be destroyed when the socket is closed - either normally or in an error situation.

Output parameters

RESPONSE
: is the domain's response to the call. It can have any of these values:
: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
: is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

. :dt.:dd.is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are: :table refid='rsntab'. :thd. :c.RESPONSE :c.Possible REASON values :ethd. :row. :c. DISASTER :c. ABEND :row. :c. EXCEPTION :c. :etable. :edl.

IIOP domain’s generic gates

Table 75 on page 936 summarizes the II domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 58. II domain’s generic gates

Gate	Trace	Function	Format
IIDM	II 0000 II 0001	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
IIST	II 0600 II 0601	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
IIXM	AP 09E0 AO 09E1	INIT_XM_CLIENT BIND_XM_CLIENT TRANSACTION_HANG ABEND_TERMINATE RELEASE_XM_CLIENT	XMAC

For descriptions of these functions and their input and output parameters, refer to the Sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—Chapter 78, “Domain manager domain (DM),” on page 663

Format STST—“System programming command flows” on page 264

Format XMAC—“Object Transaction Service domain’s specific gates” on page 939

INIT_XM_CLIENT 'saves' a pointer obtained by the IIRR socket notify gate for connection data and makes it available to the task.

The Domain Manager gates perform normal internal state initialisation and termination functions.

Modules

Module	Function
DFHIICP	II domain command processor DFHIICP provides a common mechanism for the following OT and EJ requests to call JAVA ORB code. RESYNC_COORDINATOR RESYNC_SUBORDINATE PUBLISH_CORBASERVER RETRACT_CORBASERVER PRE_INSTALL_DJAR INSTALL_DJAR DISCARD_DJAR PUBLISH_DJAR RETRACT_DJAR
DFHIIDM	Handles the following requests: PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHIIDUF	II domain offline dump formatting routine
DFHIIMM	Handles the following requests: ADD_REPLACE_RQMODEL DELETE_RQMODEL COMMIT_RQMODELS
DFHIILS	Handles the following requests via DFHIICP: ADD_LOGIGICAL_SERVER DELETE_LOGIGICAL_SERVER PUBLISH_LOGIGICAL_SERVER RETRACT_LOGIGICAL_SERVER
DFHIIRH	Handles the following requests: FIND_REQUEST_STREAM PARSE
DFHIIRP	Handles the following requests: GET_INITIAL_DATA RECEIVE_REQUEST INVOKE SEND_REPLY RECEIVE_REPLY INITIALISE TERMINATE
DFHIIRQ	Handles the following requests: INQUIRE_RQMODEL START_BROWSE GET_NEXT END_BROWSE MATCH_RQMODEL
DFHIIRR	Handles the following requests: PROCESS_REQUESTS
DFHIIST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHIITRI	Interprets II domain trace entries
DFHIIXM	Handles the following requests: INIT_XM_CLIENT BIND_XM_CLIENT TRANSACTION_HANG ABEND_TERMINATE RELEASE_XM_CLIENT

Exits

There is one user-replaceable program, DFHXOPUS, which is called by DFHIIRR during Request Receiver processing.

Trace

The point IDs for the IIOP domain are of the form IIxxxx; the corresponding trace levels are II 1, II 2 and Exc. Trace points II1000-1FFF are II JRAS trace points.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 86. Kernel domain (KE)

The kernel domain provides a consistent linkage and recovery environment for CICS.

The application programmer has no external interface to kernel linkage. However, services invoked by the application program result in execution of kernel linkage requests.

The CICS customization interface uses kernel linkage; this interface is described in the *CICS Customization Guide*.

The kernel domain, with its associated trace entries and dumped storage, becomes the first point of reference for problems that cause system recovery to be invoked. The kernel domain returns errors to the caller as response codes, if they seem to be of a form such that the caller can be expected to take alternative action.

For serious system-wide errors, the kernel domain terminates CICS with a system dump.

When the kernel domain terminates CICS following a program check or abend, messages and abend codes are produced to indicate the event that caused the kernel domain recovery routines to consider that the error was not recoverable.

Kernel domain's specific gates

Table 59 summarizes the kernel domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 59. Kernel domain's specific gates

Gate	Trace	Function	XPI
KEAR	KE 0701 KE 0702	DEREGISTER READY REGISTER WAITPRED	NO
KEDD	KE 0201 KE 0202	ADD_DOMAIN INQUIRE_DOMAIN_BY_TOKEN INQUIRE_DOMAIN_BY_NAME SET_ANCHOR INQUIRE_ANCHOR ADD_GATE DELETE_GATE INQUIRE_GLOBAL_TRACE SET_GLOBAL_TRACE INQUIRE_DOMAIN_TRACE SET_DOMAIN_TRACE INQUIRE_TASK_TRACE SET_TASK_TRACE PERFORM_SYSTEM_ACTION SET_TRAP_OFF SET_TRAP_ON SET_DEFAULT_RECOVERY	NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO

4. Only the following KEDS functions are traced:

   SEND_DEFERRED_ABEND, START_PURGE_PROTECTION, STOP_PURGE_PROTECTION, and PROCESS_KETA_ERROR.

5. The CREATE_TASK function is processed by the DFHKETA module; all other KEDS functions are processed by the DFHKEDS module.

Table 59. Kernel domain's specific gates (continued)

Gate	Trace	Function	XPI
KEDS	KE 0502	ABNORMALLY_TERMINATE_TASK	NO
	KE 0503[4]	CREATE_TASK[5]	NO
		CREATE_TCB	NO
		DETACH_TERMINATED_OWN_TCBS	NO
		END_TASK	NO
		FREE_TCBS	NO
		PUSH_TASK	NO
		POP_TASK	NO
		READ_TIME	NO
		RESET_RUNAWAY_TIMER	NO
		RESET_TIME	NO
		START_RUNAWAY_TIMER	NO
		STOP_RUNAWAY_TIMER	NO
		RESTORE_STIMER	NO
		SEND_DEFERRED_ABEND	NO
		START_PURGE_PROTECTION	YES
		STOP_PURGE_PROTECTION	YES
		START_FORCEPURGE_PROTECTION	NO
		STOP_FORCEPURGE_PROTECTION	NO
		PROCESS_KETA_ERROR	NO
KEGD	KE 0401	INQUIRE_KERNEL	NO
	KE 0402	SET_KERNEL	NO
KEIN	KE 0301	INITIALISE_KERNEL	NO
	KE 0302	SET_STATIC_TASKS	NO
		ADD_DYNAMIC_TASK	NO
		ADD_TEMPORARY_STATIC_TASK	NO
		DELETE_TASKS	NO
KETI	KE 0101	ADJUST_STCK_TO_LOCAL	NO
	KE 0102	CONVERT_TO_DECIMAL_TIME	NO
		CONVERT_TO_STCK_FORMAT	NO
		INQUIRE_DATE_FORMAT	NO
		INQ_LOCAL_DATETIME_DECIMAL	NO
		NOTIFY_RESET	NO
		REQUEST_NOTIFY_OF_A_RESET	NO
		RESET_LOCAL_TIME	NO
		SET_DATE_FORMAT	NO
KEXM	KE 0601	TRANSACTION_INITIALISATION	NO
	KE 0602		

KEAR gate, DEREGISTER function

The DEREGISTER function of the KEAR gate is used when performing a normal shutdown (and optionally at an immediate shutdown) to deregister CICS from the MVS automatic restart manager.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR|PURGED

KEAR gate, READY function

The READY function of the KEAR gate is used at the end of CICS initialization to indicate to the MVS automatic restart manager. that this CICS region is ready for work.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR|PURGED

KEAR gate, REGISTER function

The REGISTER function of the KEAR gate is used very early in CICS initialization to register CICS with the MVS automatic restart manager.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEAR gate, WAITPRED function

The WAITPRED function of the KEAR gate is used to wait on predecessors in the restart policy for this CICS region, to ensure that prerequisite subsystems are available to CICS.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR|PURGED

KEDD gate, ADD_DOMAIN function

The ADD_DOMAIN function of the KEDD gate is used to add a new domain to the domain table.

Input parameters

DOMAIN_NAME is the 8-character domain name for the new domain to be added.

DOMAIN_TOKEN is the 31-bit constant that uniquely identifies the domain, for example, DFHSM_DOMAIN for storage manager domain.

ENTRY_POINT is the 31-bit address of the entry point for that domain, for example, A(X'80000000' + DFHSMDM) for storage manager domain.

[DOMAIN_AFFINITY]
is the TCB that the domain has affinity with for TERMINATE_DOMAIN. It can have any one of these values:

STEP|RO|QR|CO|SZ

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE_DOMAIN_TOKEN DUPLICATE_DOMAIN_NAME
INVALID	INVALID_DOMAIN_TOKEN INVALID_ENTRY_POINT

KEDD gate, INQUIRE_DOMAIN_BY_TOKEN function

The INQUIRE_DOMAIN_BY_TOKEN function of the KEDD gate is used to return the domain name for a specified domain token.

Input parameters

DOMAIN_TOKEN is the 31-bit constant that uniquely identifies the domain.

Output parameters

DOMAIN_NAME is the 8-character domain name for the new domain to be added.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DOMAIN_TOKEN_NOT_FOUND
INVALID	INVALID_DOMAIN_TOKEN

KEDD gate, INQUIRE_DOMAIN_BY_NAME function

The INQUIRE_DOMAIN_BY_NAME function of the KEDD gate is used to return the domain token for a given domain name.

Input parameters

DOMAIN_NAME is the 8-character domain name for the new domain to be added.

Output parameters

DOMAIN_TOKEN is the 31-bit constant that uniquely identifies the domain.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

[REASON] is returned when RESPONSE is EXCEPTION. It has this value:

DOMAIN_NAME_NOT_FOUND

KEDD gate, SET_ANCHOR function

The SET_ANCHOR function of the KEDD gate is used to establish the calling domain's global storage pointer.

Input parameters

ANCHOR is the 31-bit address of the domain's global storage.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

[REASON] is returned when RESPONSE is INVALID. It has this value:

INVALID_DOMAIN_TOKEN

KEDD gate, INQUIRE_ANCHOR function

The INQUIRE_ANCHOR function of the KEDD gate is used to return the specified domain's global storage pointer to the caller. If the domain token is omitted, the calling domain is assumed.

Input parameters

[DOMAIN_TOKEN]
is the 31-bit constant that uniquely identifies the domain.

Output parameters

ANCHOR is the 31-bit address of the domain's global storage.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DOMAIN_TOKEN_NOT_FOUND

RESPONSE	Possible REASON values
INVALID	INVALID_DOMAIN_TOKEN

KEDD gate, ADD_GATE function

The ADD_GATE function of the KEDD gate is used to update the domain table to add a new gate to the calling domain's gate table.

Input parameters

GATE_INDEX is the 31-bit constant that uniquely identifies the gate in the domain's gate table.

ENTRY_POINT is the 31-bit address of the entry point for the gate.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE_GATE_INDEX
INVALID	INVALID_ENTRY_POINT INVALID_GATE_INDEX INVALID_DOMAIN_TOKEN

KEDD gate, DELETE_GATE function

The DELETE_GATE function of the KEDD gate is used to delete an existing gate from the calling domain's gate table.

Input parameters

GATE_INDEX is the 31-bit constant that uniquely identifies the gate in the domain's gate table.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

[REASON] When RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_GATE_INDEX INVALID_DOMAIN_TOKEN

KEDD gate, INQUIRE_GLOBAL_TRACE function

The INQUIRE_GLOBAL_TRACE function of the KEDD gate is used to return the value of the global trace flags to the caller.

Input parameters

None.

Output parameters

[MASTER_TRACE_FLAG]

determines whether tracing, for any of the trace destinations, is active. It can have either of these values:

ON|OFF

[SYSTEM_TRACE_FLAG]
determines whether tracing is allowed for tasks for which standard tracing is in effect. It can have either of these values:

ON|OFF

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDD gate, SET_GLOBAL_TRACE function

The SET_GLOBAL_TRACE function of the KEDD gate is used to store the value of the global trace flags within the kernel.

Input parameters

[MASTER_TRACE_FLAG]
determines whether tracing, for any of the trace destinations, is active. It can have either of these values:

ON|OFF

[SYSTEM_TRACE_FLAG]
determines whether tracing is allowed for tasks for which standard tracing is in effect. It can have either of these values:

ON|OFF

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDD gate, INQUIRE_DOMAIN_TRACE function

The INQUIRE_DOMAIN_TRACE function of the KEDD gate is used to return the value of the specified domain's trace flags to the caller. If the domain token is omitted, the calling domain is assumed.

Input parameters

[DOMAIN_TOKEN]
is the 31-bit constant that uniquely identifies the domain.

Output parameters

[STANDARD_TRACE_FLAGS]
is the set of 32 bits which determines selectivity of tracing within the domain for standard tasks.

[SPECIAL_TRACE_FLAGS]
is the set of 32 bits which determines selectivity of tracing within the domain for special tasks.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DOMAIN_TOKEN_NOT_FOUND
INVALID	INVALID_DOMAIN_TOKEN

KEDD gate, SET_DOMAIN_TRACE function

The SET_DOMAIN_TRACE function of the KEDD gate is used to store the value of the specified domain's trace flags in the kernel. If the domain token is omitted, the calling domain is assumed.

The current task's stack entries are updated to reflect the change. The trace count is incremented so that all other tasks have their stack entries refreshed when they are next dispatched.

Input parameters

[DOMAIN_TOKEN]
is the 31-bit constant that uniquely identifies the domain.

[STANDARD_TRACE_FLAGS]
is the set of 32 bits which determines selectivity of tracing within the domain for standard tasks.

[SPECIAL_TRACE_FLAGS]
is the set of 32 bits which determines selectivity of tracing within the domain for special tasks.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|DISASTER`

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DOMAIN_TOKEN_NOT_FOUND
INVALID	INVALID_DOMAIN_TOKEN

KEDD gate, INQUIRE_TASK_TRACE function

The INQUIRE_TASK_TRACE function of the KEDD gate is used to return the value of the calling task's trace flag to the caller.

Input parameters

None.

Output parameters

[TRACE_TYPE]
determines whether standard, special, or no tracing is required for this task. It can have any one of these values:

`STANDARD|SPECIAL|SUPPRESSED`

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|DISASTER`

KEDD gate, SET_TASK_TRACE function

The SET_TASK_TRACE function of the KEDD gate is used to store the value of the task trace flag in the current task's task table[6] entry.

The current task's stack entries are updated to reflect the change.

Input parameters

TRACE_TYPE
determines whether standard, special, or no tracing is required for this task. It can have any one of these values:

`STANDARD|SPECIAL|SUPPRESSED`

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|DISASTER`

6. **Task table**: A task table is a logical block of tasks, allocated together by the Kernel domain, and used to simplify the process of dynamically adding new tasks. Task tables are chained together, and vary in number.

KEDD gate, PERFORM_SYSTEM_ACTION function

The PERFORM_SYSTEM_ACTION function of the KEDD gate is used in exceptional circumstances either to terminate CICS (with or without a dump) or to take an MVS SDUMP.

Normally, these services are invoked from domains during preinitialization before the dump domain is available.

Input parameters

[TERMINATE_SYSTEM (YES, NO)]
: specifies whether CICS is to be terminated or not. It can have either of these values:

  YES|NO

[DUMP_SYSTEM (YES, NO)]
: specifies whether an MVS SDUMP is to be taken or not. It can have either of these values:

  YES|NO

[NORMAL_TERMINATION(YES, NO)]
: specifies whether CICS is being terminated normally. Normal termination includes controlled and immediate shutdowns. It can have either of these values:

  YES|NO

  The default value is NO.

Output parameters

RESPONSE
: is the domain's response to the call. It can have any of these values:

  OK|EXCEPTION|INVALID|DISASTER

KEDD gate, SET_TRAP_OFF function

The SET_TRAP_OFF function of the KEDD gate is used to reset the kernel global trap point.

Input parameters

None.

Output parameters

RESPONSE
: is the domain's response to the call. It can have any of these values:

  OK|EXCEPTION|INVALID|DISASTER

KEDD gate, SET_TRAP_ON function

The SET_TRAP_ON function of the KEDD gate is used to set a kernel global trap point.

Input parameters

ENTRY_POINT
: is the 31-bit address of the kernel global trap.

Output parameters

RESPONSE
: is the domain's response to the call. It can have any of these values:

  OK|EXCEPTION|INVALID|DISASTER

[REASON]
: is returned when RESPONSE is INVALID. It has this value:

  INVALID_ENTRY_POINT

KEDD gate, SET_DEFAULT_RECOVERY function

The SET_DEFAULT_RECOVERY function of the KEDD gate is used to establish the calling domain's default recovery routine. Used by the Application domain to identify DFHSRP as its default recovery routine.

Input parameters

ENTRY_POINT is the 31-bit address of the entry point for the recovery routine.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

[REASON] is returned when RESPONSE is INVALID. It has this value:

INVALID_DOMAIN_TOKEN

KEDS gate, ABNORMALLY_TERMINATE_TASK function

The ABNORMALLY_TERMINATE_TASK function of the KEDS gate identifies the task which is to be abnormally terminated.

Input parameters

TASK_TOKEN identifies the task which is to be abnormally terminated.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TERMINATE_FAILED

KEDS gate, CREATE_TASK function

The CREATE_TASK function of the KEDS gate is used to allocate a new executable task from the task table[7].

When the task is first dispatched, the Kernel domain issues a KEDS_TASK_REPLY request, which passes control to the Dispatcher domain's task reply gate. (See "KEDS format, TASK_REPLY function" on page 849.)

The attach token input on the CREATE_TASK request is passed back to the dispatcher domain on the TASK_REPLY, to identify the CREATE_TASK and TASK_REPLY pair.

Note: The CREATE_TASK function is processed by the DFHKETA module.

Input parameters

ALLOCATION indicates whether or not the returned task should be allocated from those tasks pre-allocated for MXT. It can either of these values:

STATIC | DYNAMIC

ATTACH_TOKEN is the 31-bit token that uniquely identifies the request. This token is returned on the corresponding TASK_REPLY to identify the request.

Output parameters

TASK_TOKEN is the 31-bit token that uniquely identifies the newly created task.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

7. **Task table**: A task table is a logical block of tasks, allocated together by the Kernel domain, and used to simplify the process of dynamically adding new tasks. Task tables are chained together, and vary in number.

RESPONSE	Possible REASON values
DISASTER	INQUIRE_ERROR
EXCEPTION	ADD_TASK_ERROR

KEDS gate, CREATE_TCB function

The CREATE_TCB function of the KEDS gate creates the default task for a new MVS TCB, and MVS posts the TCB to start execution. The default task is the task, associated with the TCB, that executes the dispatcher loop which chooses the next CICS task (system or non-system) to be dispatched, or if no CICS task is to be dispatched, issues an MVS WAIT.

The kernel invokes the dispatcher domain at its KEDS gate with a TCB_REPLY request, under the new TCB's default task.

The attach token is used to identify the CREATE_TCB and TCB_REPLY pair.

Input parameters

ATTACH_TOKEN
is the 31-bit token that uniquely identifies the request. This token is returned on the corresponding TCB_REPLY to identify the request.

DEPENDENT_ON
specifies that the TCB is dependent on the named parent TCB mode. This parameter is used to ensure that in the case of an immediate shutdown, worker JVMs (which are built on J8 or J9 mode TCBs) are terminated before master JVMs (which are built on JM mode TCBs).

ESSENTIAL_TCB
indicates whether CICS is to be terminated if a TCB in this mode has its ESTAE exit driven for a non recoverable error.

EXEC_CAPABLE
indicates whether support should be provided under the new TCB for CICS API commands.

INHERIT_SUBSPACE
indicates whether TCBs in this mode are to inherit the subspace of the attaching TCB.

LE_ENVIRONMENT
indicates whether CICS should tell Language Environment that it is running in a CICS environment under this TCB. If LE_CICS is specified, Language Environment will issue CICS API commands.

[MODE]
specifies the mode of the new TCB. It can have any one of these values:

RO|QR|CO|SZ|RP|FO

PARENT_MODENAME
identifies the mode of the TCB that is to ATTACH the new TCB.

PRTY_RELATIVE_TO_QR
gives the priority of this TCB relative to QR.

[SZERO]
gives the value (YES or NO) of the SZERO parameter for the ATTACH request. If TCB_KEY(USERKEY) is specified, SZERO(NO) is assumed.

TCB_KEY
specifies the key to be specified on the ATTACH of TCBs in this mode. The value ends up in TCBPKF. It can have either of these values:

CICSKEY|USERKEY

Output parameters

TASK_TOKEN
is the 31-bit token that uniquely identifies the new TCB's task.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	INQUIRE_ERROR

RESPONSE	Possible REASON values
EXCEPTION	ADD_TASK_ERROR ADD_TCB_ERROR ATTACH_KTCB_ERROR

KEDS gate, DETACH_TERMINATED_OWN_TCBS function

The DETACH_TERMINATED_OWN_TCBS function of the KEDS gate detaches any terminated TCBs which were attached by the TCB on which this function is invoked.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have this value:

OK

KEDS gate, END_TASK function

The END_TASK function of the KEDS gate is used to free any resources that have been acquired by the kernel domain during the lifetime of the current task and need freeing before the end of the task.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any one of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, FREE_TCBS function

The FREE_TCBS function of the KEDS gate conditionally frees control blocks, in collaboration with the Dispatcher for re-use, associated with any detached TCBs.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have this value:

OK

KEDS gate, PUSH_TASK function

Given a TCB executing its default task, the PUSH_TASK function of the KEDS gate is used to make it execute a CICS task instead.

Input parameters

TASK_TOKEN is the 31-bit token that identifies the CICS task to be executed.

Output parameters

[INTERVAL] is a doubleword containing the CPU time used by the task while it was pushed.

RESPONSE is the domain's response to the call. It can have any one of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, POP_TASK function

Given a TCB executing the current CICS task, the POP_TASK function of the KEDS gate is used to make it execute its default task instead.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, READ_TIME function

The READ_TIME function of the KEDS gate is used to obtain the total CPU time that the current task has taken so far and the accumulated CPU time for the current TCB.

Input parameters

None.

Output parameters

[INTERVAL] A doubleword containing the total CPU time used so far.

[ACCUM_TIME] A doubleword containing the accumulated CPU time used so far by the current TCB.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, RESET_RUNAWAY_TIMER function

The RESET_RUNAWAY_TIMER function of the KEDS gate is used to reset runaway timing for the current
task. This resets the count of outstanding STOP_RUNAWAY_TIMER requests.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, RESET_TIME function

The RESET_TIME function of the KEDS gate is used to reset the total CPU time that the current task has taken so far.

Input parameters

None.

Output parameters

[INTERVAL] A doubleword containing the total CPU time used so far.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, STOP_RUNAWAY_TIMER function

The STOP_RUNAWAY_TIMER function of the KEDS gate is used to inhibit runaway detection for the current task. The remaining runaway interval is preserved until a START_RUNAWAY_TIMER request is issued. The stop runaway count is incremented by one; this allows STOP_RUNAWAY_TIMER requests to be nested.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, START_RUNAWAY_TIMER function

The START_RUNAWAY_TIMER function of the KEDS gate is used to resume runaway timing for the current task. This reduces the stop runaway count by one. The timer is resumed only when all outstanding STOP_RUNAWAY_TIMER requests have been canceled.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, RESTORE_STIMER function

The RESTORE_STIMER function of the KEDS gate is used to restore the kernel's STIMER exit after MVS requests that use the MVS STIMER macro internally.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, SEND_DEFERRED_ABEND function

The SEND_DEFERRED_ABEND function of the KEDS gate is used by the transaction manager to implement the deferred purge function. If a purge request is made against a task that is not in a suitable state to be purged, this function defers the abend of that task until the task is no longer protected against purge.

This function is used by the transaction manager to implement the deferred purge function.

Input parameters

[DS_TASK_TOKEN]
is the 31-bit dispatcher token that identifies the CICS task to be abended. If not supplied, DS_TASK_TOKEN defaults to the current task.

ABEND_CODE
is the four-character abend code for the abend.

[FORCE]
indicates whether or not the deferred abend is to be forced. It can have either of these values:

YES|NO

The default is NO.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, START_PURGE_PROTECTION function

The START_PURGE_PROTECTION function of the KEDS gate is used to inhibit purge, but not force-purge, for the current task.

In general, each START_PURGE_PROTECTION call should have a corresponding STOP_PURGE_PROTECTION function call to end the purge protection period on completion of any program logic that needs such protection.

This function increments by one the purge protection count for the task. You can issue several START_PURGE_PROTECTION commands for the same task, to increase the count for the task. (To enable the task to purged, the count must be decremented to zero by issuing STOP_PURGE_PROTECTION commands.)

Input parameters
None.

Output parameters
RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, STOP_PURGE_PROTECTION function
The STOP_PURGE_PROTECTION function of the KEDS gate is used to enable again purge for the current task after purge has been suspended by a previous START_PURGE_PROTECTION function call.

This function decrements by one the purge protection count for the task. To enable the task to purged, the count must be decremented to zero by issuing the appropriate number of STOP_PURGE_PROTECTION commands.

You must design your exit programs to ensure that purge protection is correctly cancelled. For more information about using these functions to stop and start purge protection, see the *CICS Customization Guide*.

Input parameters
None.

Output parameters
RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, START_FORCEPURGE_PROTECTION function
The START_FORCEPURGE_PROTECTION function of the KEDS gate is used by CICS to inhibit forcepurge for a task. The STOP_FORCEPURGE_PROTECTION function is used to end the forcepurge protection period.

Input parameters
None.

Output parameters
RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, STOP_FORCEPURGE_PROTECTION function
The STOP_FORCEPURGE_PROTECTION function of the KEDS gate is used by CICS to enable again forcepurge for a task after forcepurge has been suspended by a previous START_FORCEPURGE_PROTECTION function call.

Input parameters
None.

Output parameters
RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEDS gate, PROCESS_KETA_ERROR function

The PROCESS_KETA_ERROR function of the KEDS gate is used to handle any errors for the DFHKETA module. (The DFHKETA module handles the performance sensitive KEDS functions, and calls the DFHKEDS module when its recovery routine is invoked.)

Input parameters

ERROR_DATA
address of the error data that describes the error that has occurred in the DFHKETA module.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KEGD gate, INQUIRE_KERNEL function

The INQUIRE_KERNEL function of the KEGD gate is used to obtain the global data maintained by the kernel.

Input parameters

None.

Output parameters

[CICS_SVC_NUMBER]
is the 8-bit CICS service SVC number.

[SPECIFIC_APPLID]
is the 8-character specific applid that identifies the CICS system in the VTAM network.

[GENERIC_APPLID]
is the 8-character generic applid that identifies the active and alternate CICS systems to VTAM in an XRF environment.

[XRF_COMMAND_LIST]
is the 8-character name of the command list table used by the XRF alternate CICS region.

[ALTERNATE_XRF_IDS]
is the 8-character name of the recoverable service table used if the CICS region is running with XRF and DBCTL.

[SYSID]
is the 4-character ZCP system entry name.

[SIT_NAME]
is the 8-character SIT name.

[OS_PARMS]
is the 8-byte block containing the 31-bit address and 31-bit length of the MVS parameters.

[OP_SYS]
is the 1-character operating system identifier, for example, 'B' = MVS.

[OP_REL]
is the 2-byte operating system release and modification level.

[HPO]
specifies whether CICS is to use the VTAM high performance option. It can have either of these values:

YES|NO

[SYSTEM_RUNAWAY_LIMIT]
the ICVR time to be used by all tasks that have been defined to have the default runaway limit in the system.

[CPU_MONITORING]
specifies whether the kernel is to perform CPU monitoring. It can have either of these values:

YES|NO

[USS_AVAILABLE]
specifies whether the kernel successfully issued a Unix System Services (USS) SET_DUB_DEFAULT DUBPROCESS command during CICS initialization. It can have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

KEGD gate, SET_KERNEL function

The SET_KERNEL function of the KEGD gate is used to change the global data maintained by the kernel.

Input parameters

[CICS_SVC_NUMBER]
is the 8-bit CICS service SVC number.

[SPECIFIC_APPLID]
is the 8-character specific applid that identifies the CICS system in the VTAM network.

[GENERIC_APPLID]
is the 8-character generic applid that identifies the active and alternate CICS systems to VTAM in an XRF environment.

[XRF_COMMAND_LIST]
is the 8-character name of the command list table used by the XRF alternate CICS region.

[ALTERNATE_XRF_IDS]
is the 8-character name of the recoverable service table used if the CICS region is running with XRF and DBCTL.

[SYSID]
is the 4-character ZCP system entry name.

[SIT_NAME]
is the 8-character name of the system initialization table.

[HPO]
specifies whether CICS is to use the VTAM high performance option. It can have either of these values:

YES|NO

[SYSTEM_RUNAWAY_LIMIT]
the ICVR time to be used by all tasks that have been defined to have the default runaway limit in the system.

[CPU_MONITORING]
specifies whether the kernel is to perform CPU monitoring. It can have either of these values:

YES|NO

[TERMINATE_FO]
specifies whether the FO TCB can be normally terminated on an immediate shutdown.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

KETI gate, RESET_LOCAL_TIME function

The RESET_LOCAL_TIME function of the KETI gate is used by the AP domain to inform KETI that a local time reset has occurred.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any one of these values:

OK|INVALID|KERNERROR|PURGED|DISASTER

KETI gate, REQUEST_NOTIFY_OF_A_RESET function

The REQUEST_NOTIFY_OF_A_RESET function of the KETI gate requests a shoulder tap from KETI whenever the local time is reset.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any one of these values:

OK|INVALID|KERNERROR|PURGED|DISASTER

KETI gate, SET_DATE_FORMAT function

The SET_DATE_FORMAT function of the KETI gate is used to set the date format for the timer domain.

Input parameters

DATE_FORMAT
is the format to be set as the default for the timer domain. It can have any one of these values:

YYMMDD|DDMMYY|MMDDYY

Output parameters

RESPONSE
is the domain's response to the call. It can have any one of these values:

OK|INVALID|KERNERROR|PURGED|DISASTER

KETI gate, INQUIRE_DATE_FORMAT function

The INQUIRE_DATE_FORMAT function of the KETI gate is used to return the current date format.

Input parameters

None.

Output parameters

DATE_FORMAT
is the current default date format for the timer domain. It can have any one of these values:

YYMMDD|DDMMYY|MMDDYY

RESPONSE
is the domain's response to the call. It can have any one of these values:

OK|INVALID|KERNERROR|PURGED|DISASTER

KETI gate, INQ_LOCAL_DATETIME_DECIMAL function

The INQ_LOCAL_DATETIME_DECIMAL function of the KETI gate is used to return the local date, and the local time in decimal format.

Input parameters

None.

Output parameters

DECIMAL_DATE
is an 8-character date in the format determined by FULL_DATE_FORMAT.

DECIMAL_TIME
is the current local decimal time in the format HHMMSS.

DECIMAL_MICROSECONDS
is the 6-character microseconds portion of DECIMAL_TIME.

FULL_DATE_FORMAT
is the current full date format determined by the default date format of the timer domain. It can have any one of these values:

YYYYMMDD|DDMMYYYY|MMDDYYYY

RESPONSE
is the domain's response to the call. It can have any one of these values:

OK|INVALID|KERNERROR|PURGED|DISASTER

KETI gate, CONVERT_TO_DECIMAL_TIME function

The CONVERT_TO_DECIMAL_TIME function of the KETI gate is used to convert dates and times in the internal store clock (STCK) format to decimal format.

Input parameters

STCK_TIME
is a doubleword containing a date and time in STCK format.

Output parameters

DECIMAL_DATE
is an 8-character date in the format determined by FULL_DATE_FORMAT

DECIMAL_TIME is the current local decimal time in the format HHMMSS

DECIMAL_MICROSECONDS
is the 6-character microseconds portion of DECIMAL_TIME

FULL_DATE_FORMAT
is the current full date format determined by the default date format of the timer domain. It can have any one of these values:

YYYYMMDD|DDMMYYYY|MMDDYYYY

RESPONSE is the domain's response to the call. It can have any one of these values:

OK|INVALID|KERNERROR|PURGED|DISASTER

KETI gate, CONVERT_TO_STCK_FORMAT function

The CONVERT_TO_STCK_FORMAT function of the KETI gate is used to convert times and dates to STCK format.

Input parameters

DECIMAL_TIME is the current local decimal time in the format HHMMSS.

[DECIMAL_DATE]
is an optional 8-character date in the format determined either by FULL_DATE_FORMAT or by the default for the timer domain if FULL_DATE_FORMAT is omitted.

[INSTANCE] is required only if DECIMAL_DATE is omitted. It can have either of these values:

LAST|NEXT

[FULL_DATE_FORMAT]
is the current full date format. It can have any one of these values:

YYYYMMDD|DDMMYYYY|MMDDYYYY

Output parameters

STCK_TIME is a doubleword containing the GMT STCK value corresponding to the input local time.

RESPONSE is the domain's response to the call. It can have any one of these values:

OK|INVALID|KERNERROR|PURGED|DISASTER

KEXM gate, TRANSACTION_INITIALISATION function

The TRANSACTION_INITIALISATION function of the KEXM gate is used to perform kernel initialisation during XM task-reply.

Input parameters

TRANSACTION_TOKEN
is a token identifying the transaction for which kernel initialization is to be performed.

Output parameters

RESPONSE is the domain's response to the call. It can have any one of these values:

OK|INVALID|KERNERROR|PURGED|DISASTER

Kernel domain's generic formats

Table 60 describes the generic formats owned by the kernel domain, and shows the functions performed on the calls.

Table 60. Generic formats owned by the kernel domain

Format	Calling module	Function
KEDS	DFHKETA DFHKETCB	TASK_REPLY TCB_REPLY
KETI	DFHKETI	NOTIFY_RESET

In the descriptions of the formats that follow, the "input" parameters are input not to the kernel domain, but to the domain being called by the kernel domain. Similarly, the "output" parameters are output by the domain that was called by the kernel domain, in response to the call.

KEDS format, TASK_REPLY function

The TASK_REPLY function of the KEDS format is issued by the kernel to the issuer of CREATE_TASK, under the new task.

Input parameters

ATTACH_TOKEN is the 31-bit token that uniquely identifies the corresponding CREATE_TASK request.

TASK_TOKEN is the 31-bit token that uniquely identifies the new task.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|INVALID|KERNERROR|PURGED|DISASTER

KEDS format, TCB_REPLY function

The TCB_REPLY function of the KEDS format is issued by the kernel to the issuer of CREATE_TCB, under the new TCB's default task.

Input parameters

ATTACH_TOKEN is the 31-bit token that uniquely identifies the corresponding CREATE_TCB request.

TASK_TOKEN is the 31-bit token that uniquely identifies the new TCB's task.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

KETI format, NOTIFY_RESET function

The NOTIFY_RESET function of the KETI format is used by KETI itself to inform domains that a RESET has occurred.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any one of these values:

OK|KERNERROR|PURGED|DISASTER

Control blocks

Figure 110 on page 850 shows the MVS TCB structure used by CICS. Other TCBs are attached under the quasi-reentrant TCB by IBM DATABASE 2 (DB2) or IMS code, if those products are being used.

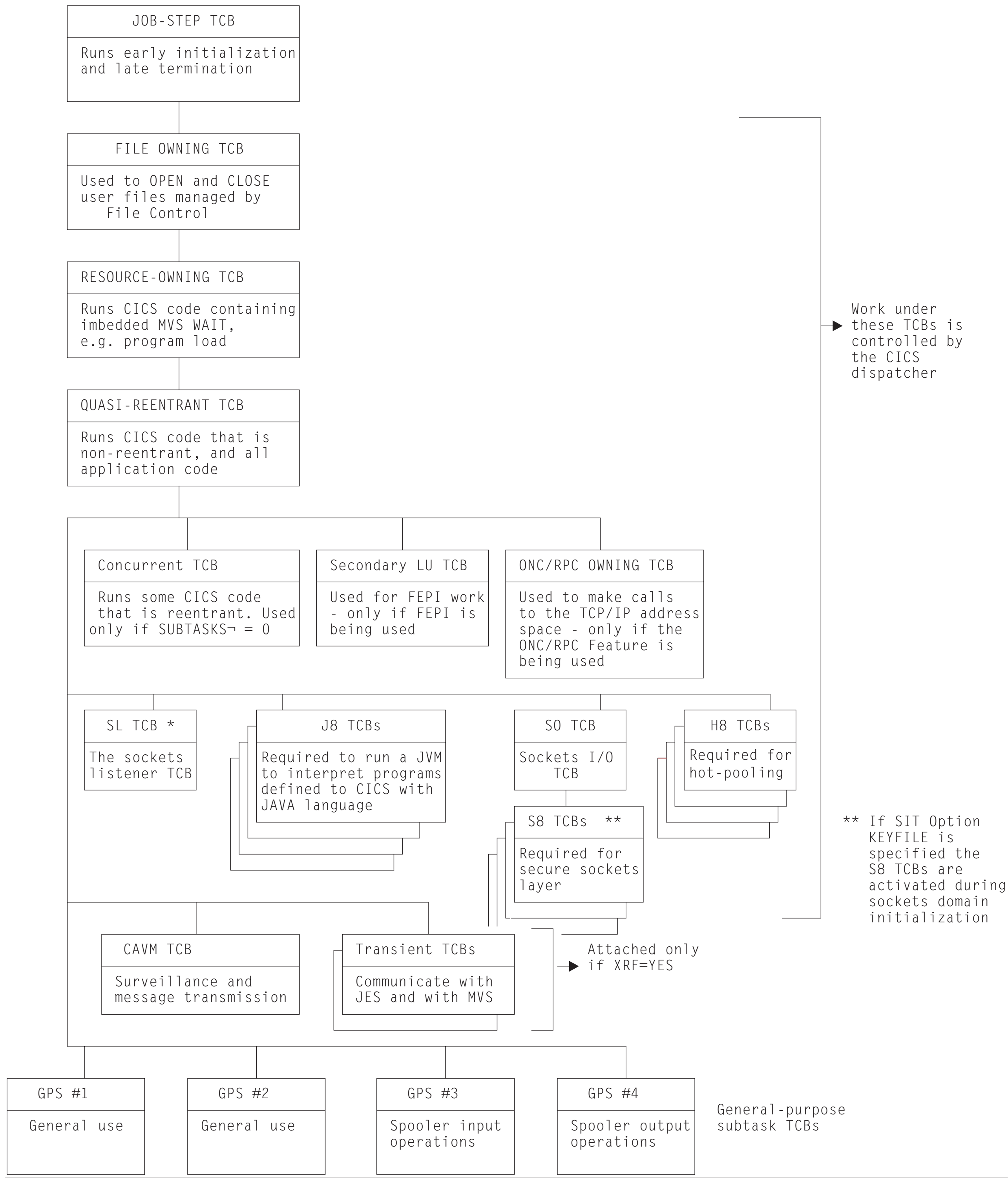


Figure 110. MVS TCB structure used by CICS

Modules

Module	Function
DFHKEAR	Implements KEAR service requests.
DFHKEDCL	Implements domain call requests.
DFHKEDD	Services KEDD-format requests.
DFHKEDRT	Implements domain return requests.
DFHKEDS	Services KEDS-format requests.
DFHKEDUF	Offline dump formatting routine to format the kernel domain control blocks.
DFHKEEDA	Handles deferred abends
DFHKEGD	Services KEGD-format requests.
DFHKEIN	Implements kernel domain initialization.
DFHKELCL	Implements LIFO Push.
DFHKELOC	Offline dump formatting routine to locate the kernel domain anchor blocks.
DFHKELRT	Implements LIFO Pop.
DFHKERCD	Constructs the kernel domain error data for error handling routines.
DFHKERER	Updates the kernel domain error table for error handling routines.
DFHKERET	Implements RESET_ADDRESS requests.
DFHKERKE	Handles KERNERROR responses for domain call requests which cannot handle them.
DFHKERPC	Implements recovery percolation both from RECOVERY_PERCOLATE requests and also other recovery events that, because of the existing environment, must be percolated.
DFHKERRI	Responsible for actually passing control to a recovery routine.
DFHKERRQ	Implements RECOVERY_REQUEST requests.
DFHKERRU	Implements runaway task error handling.
DFHKERRX	Implements RECOVERY_EXIT requests.
DFHKESCL	Implements subroutine call requests.
DFHKESFM	Handles freeing of stack segments.
DFHKESGM	Handles allocation of new stack segments.
DFHKESIP	Receives control from and returns control to MVS.
DFHKESRT	Implements subroutine return requests.
DFHKESTX	Is the CICS ESTAE exit and passes control to the appropriate level of recovery routine.
DFHKESVC	Provides authorised services for kernel domain functions.
DFHKETA	Implements KEDS CREATE_TASK requests.
DFHKETCB	Receives control from MVS for a kernel domain TCB.
DFHKETI	Provides service time functions at the KETI gate.
DFHKETIX	Performs task CPU monitoring functions and task runaway detection.
DFHKETRI	Offline trace formatting routine for kernel domain trace entries.
DFHKETXR	Allows an attaching TCB to detmine that a TCB (but not a specific TCB) which it attached, has terminated. This allows for the possibility of initiating a more timely detach of TCBs which have terminated normally, and to detect TCBs which have prematurely terminated.
DFHKEXM	Implements KEXM_FORMAT requests.

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the kernel domain are of the form KE xxxx; the corresponding trace levels are KE 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 87. Loader domain (LD)

The loader domain is used by the domains of the CICS system to obtain access to storage-resident copies of nucleus and application programs, maps, and tables. In order to provide this, the loader domain interfaces with MVS to perform the loading of programs into the CICS dynamic storage areas (DSAs), and the scanning of the MVS link pack area (LPA).

The most common functions provided by the loader domain are:

ACQUIRE_PROGRAM
: used to obtain the load point and entry point addresses and length of a usable program copy, and to reserve the copy for use by the caller.

RELEASE_PROGRAM
: used to inform the loader domain that a specific program copy is no longer required.

DEFINE_PROGRAM
: used to inform the loader domain of the CICS attributes of a program.

REFRESH_PROGRAM
: used to request the loader domain to rescan the LPA or DFHRPL library for a new copy of a program.

The loader domain is utilized by many domains in the system, but its most common user is the program manager domain, for access to application programs. The program manager domain issues the following requests:

ACQUIRE_PROGRAM
: whenever a program issues a LINK, XCTL, or LOAD command to link to, transfer control to, or load another program.

DEFINE_PROGRAM
: as part of a request to define or autoinstall a program, mapset, or partitionset.

RELEASE_PROGRAM
: whenever a called program issues a RETURN command to return control to the calling program, or a program issues a RELEASE command to release a loaded program.

REFRESH_PROGRAM
: as part of an EXEC CICS SET PROGRAM NEWCOPY or PHASEIN request.

Loader domain's specific gate

Table 61 summarizes the loader domain's specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 61. Loader domain's specific gate

Gate	Trace	Function	XPI
LDLD	LD 0001	ACQUIRE_PROGRAM	YES
	LD 0002	RELEASE_PROGRAM	YES
		REFRESH_PROGRAM	NO
		DEFINE_PROGRAM	YES
		INQUIRE_PROGRAM	NO
		DELETE_PROGRAM	YES
		START_BROWSE	NO
		GET_NEXT_PROGRAM	NO
		GET_NEXT_INSTANCE	NO
		END_BROWSE	NO
		IDENTIFY_PROGRAM	NO
		SET_OPTIONS	NO
		INQUIRE_OPTIONS	NO
		CATALOG_PROGRAMS	NO

LDLD gate, ACQUIRE_PROGRAM function

The ACQUIRE_PROGRAM function of the LDLD gate is used to obtain the entry point and load point addresses and the length of a usable copy of the named program. The program must previously have been identified to the system in a DEFINE request, either during this session or in a previous session, if the catalog is in use.

Input parameters

PROGRAM_NAME
specifies the name of the required program.

PROGRAM_TOKEN
is a valid program-identifying token as returned by a previous DEFINE or ACQUIRE request for the same program name.

[SUSPEND]
indicates whether the caller expects to receive control with an exception response if the loader encounters a shortage of virtual storage, or other transient error conditions. It can have either of these values:

YES|NO

If there is insufficient storage to satisfy the request, SUSPEND(YES) causes the caller to be suspended until the request can be satisfied, and SUSPEND(NO) causes an exception response (reason NO_STORAGE) to be returned to the caller.

Output parameters

ENTRY_POINT
is the address of the entry point of the program instance.

[LOAD_POINT]
is the address of the load point of the program instance.

[PROGRAM_LENGTH]
is the length of the program instance in bytes.

[NEW_PROGRAM_TOKEN]
is the identifying token that may be used on subsequent ACQUIRE or RELEASE calls for this program name.

[PROGRAM_ATTRIBUTE]
reflects the program attribute from the program definition, and is used by the program manager domain to recognize RELOAD programs.

[LOCATION]
determines where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

[COPY_STATUS]
indicates whether this request resulted in a physical load of the program into storage, and is used by the program manager domain to recognize that a COBOL program requires initialization.

[FETCH_TIME]
is the time taken to load the program from the DFHRPL library. This is represented as the middle 4 bytes of a doubleword stored clock (STCK) value. If the acquired program resides in the MVS link pack area (LPA) or has already been loaded into one of the CICS dynamic storage areas (DSAs), the returned value is zero.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LIBRARY_IO_ERROR OS_STORAGE_SHORTAGE ABEND LOOP
EXCEPTION	PROGRAM_NOT_DEFINED PROGRAM_NOT_FOUND NO_STORAGE
INVALID	INVALID_PROGRAM_TOKEN

LDLD gate, RELEASE_PROGRAM function

The RELEASE_PROGRAM function of the LDLD gate is used to inform the loader domain that use of a copy of the named program is no longer required. The use count of the specified program instance is decremented; if the use count reaches zero, and the program is eligible to be removed from memory, it is removed from memory.

Input parameters

PROGRAM_NAME specifies the name of the program to be released.

PROGRAM_TOKEN
is the identifying token returned by the ACQUIRE request for this program.

ENTRY_POINT specifies the address of the entry point of the module.

Output parameters

[LOAD_POINT] is the address of the load point of the program instance.

[PROGRAM_LENGTH]
is the length of the program instance in bytes.

[LOCATION] determines where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	PROGRAM_NOT_DEFINED PROGRAM_NOT_IN_USE
INVALID	INVALID_PROGRAM_TOKEN INVALID_ENTRY_POINT

LDLD gate, REFRESH_PROGRAM function

The REFRESH_PROGRAM function of the LDLD gate is used to inform the loader domain that a new version of the program has been cataloged, and that this version of the named program should be used for all future ACQUIRE requests.

Input parameters

PROGRAM_NAME specifies the name of the program that is to have a new version used.

Output parameters

[NEW_VERSION_FOUND]
indicates whether a new version of the program has been found.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LIBRARY_IO_ERROR OS_STORAGE_SHORTAGE ABEND LOOP

RESPONSE	Possible REASON values
EXCEPTION	PROGRAM_NOT_DEFINED PROGRAM_NOT_FOUND

LDLD gate, DEFINE_PROGRAM function

The DEFINE_PROGRAM function of the LDLD gate is used to introduce a new program to the CICS system or to update the details of an existing program.

Input parameters

PROGRAM_NAME
specifies the name of the program whose attributes are to be set.

CATALOG_MODULE
indicates whether the program definition should be written to one of the catalogs. It can have either of these values:

YES|NO

UPDATE
indicates whether the loader domain should update the program definition if the loader domain already has a program definition for the program. If UPDATE(NO) is specified, and the loader domain already has a program definition for the specified program, PROGRAM_ALREADY_DEFINED is returned. It can have either of these values:

YES|NO

[EXECUTION_KEY]
is the execution key for the program. This is used to determine which DSA the program instance resides in. It can have either of these values:

USER|CICS

[PROGRAM_TYPE]
is the type of program copy to be used. It can have any of these values:

PRIVATE|SHARED|TYPE_ANY

[PROGRAM_USAGE]
defines whether the program is part of the CICS nucleus, or is an application program defined by the user. This determines whether the program definition is written to the local catalog or to the global catalog. It can have either of these values:

NUCLEUS|APPLICATION

[PROGRAM_ATTRIBUTE]
is a residency attribute to be associated with the program. It can have any of these values:

RESIDENT|REUSABLE|TRANSIENT|RELOAD

[REQUIRED_AMODE]
is the addressing mode required by CICS for the program. A program that does not have the required residency mode is not loaded. It can have any of these values:

24|31|AMODE_ANY

[REQUIRED_RMODE]
is the residency mode required by CICS for the program. A program that does not have the required mode requirements is not loaded. It can have any of these values:

24|RMODE_ANY

Output parameters

[NEW_PROGRAM_TOKEN]
is an identifying token that can be used on subsequent ACQUIRE or RELEASE calls for this program name.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	CATALOG_NOT_OPERATIONAL CATALOG_ERROR INVALID_PROGRAM_NAME PROGRAM_ALREADY_DEFINED
INVALID	INVALID_MODE_COMBINATION INVALID_TYPE_ATTRIB_COMBIN

LDLD gate, INQUIRE_PROGRAM function

The INQUIRE_PROGRAM function of the LDLD gate is used to return the details of a specific program.

Input parameters

PROGRAM_NAME specifies the name of the program whose attributes are being requested.

PROGRAM_TOKEN
is a valid program token as returned by a previous DEFINE or ACQUIRE request, or obtained from the PPT entry, for the program.

Output parameters

[NEW_PROGRAM_TOKEN]
is an identifying token that can be used on subsequent ACQUIRE or RELEASE calls for this program name.

[PROGRAM_TYPE]
is the current program copy type.

[PROGRAM_USAGE]
is the current usage definition.

[EXECUTION_KEY]
is the execution key for the program.

[PROGRAM_ATTRIBUTE]
is the current residency attribute of the program.

[SPECIFIED_AMODE]
is the addressing mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_AMODE was omitted when the program was defined, AMODE_NOT_SPECIFIED is returned.

[SPECIFIED_RMODE]
is the residency mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_RMODE was omitted when the program was defined, RMODE_NOT_SPECIFIED is returned.

[PROGRAM_LENGTH]
is the length of the program in bytes. If the program has not been used, this is zero.

[PROGRAM_USE_COUNT]
is the cumulative use count of the program.

[PROGRAM_USER_COUNT]
is the current number of users of the program.

[LOAD_POINT] is the address of the load point of the last program instance created for this program name.

[ENTRY_POINT]
is the address of the entry point of the last program instance created for this program name.

[LOCATION] indicates where the program for which the LOAD_POINT and ENTRY_POINT have been returned resides.

[ACCESS] is the type of storage that the program resides in.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	PROGRAM_NOT_DEFINED
INVALID	INVALID_PROGRAM_TOKEN

LDLD gate, DELETE_PROGRAM function

The DELETE_PROGRAM function of the LDLD gate is used to remove a program from the CICS system. All subsequent ACQUIRE requests for the named program fail with a reason of PROGRAM_NOT_DEFINED. Any instance of the program in use at the time the DELETE is received continue to exist until a RELEASE request reduces the use count to zero, at which time the instance is removed from memory.

Input parameters

PROGRAM_NAME specifies the name of the program to be removed.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	PROGRAM_NOT_DEFINED

LDLD gate, START_BROWSE function

The START_BROWSE function of the LDLD gate is used to start a browse session.

Input parameters

[PROGRAM_NAME]
specifies the name of the program whose attributes are to be returned.

[ENTRY_POINT]
is the address of the entry point of the last program instance created for this program name.

Output parameters

BROWSE_TOKEN is a token used to refer to this browse session on subsequent browse requests.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

LDLD gate, GET_NEXT_PROGRAM function

The GET_NEXT_PROGRAM function of the LDLD gate is used to perform an INQUIRE function for the next program in the alphabetic sequence of programs in the current browse session.

Input parameters

BROWSE_TOKEN is a valid browse token as returned by the preceding START_BROWSE request.

Output parameters

[PROGRAM_NAME]
is the name of the program whose attributes have been returned.

[PROGRAM_TYPE]
is the current program copy type.

[PROGRAM_USAGE]
is the current usage definition.

[EXECUTION_KEY]
is the execution key for the program.

[PROGRAM_ATTRIBUTE]
is the current residency attribute of the program.

[SPECIFIED_AMODE]
is the current addressing mode required by CICS for the program. If REQUIRED_AMODE was omitted when the program was defined, AMODE_NOT_SPECIFIED is returned.

[SPECIFIED_RMODE]
is the current residency mode required by CICS for the program. If REQUIRED_RMODE was omitted when the program was defined, RMODE_NOT_SPECIFIED is returned.

[PROGRAM_LENGTH]
is the length of the program in bytes. If the program has not been used, this is zero.

[PROGRAM_USE_COUNT]
is the cumulative use count of the program.

[PROGRAM_USER_COUNT]
is the current number of users of the program.

[LOAD_POINT] is the address of the load point of the last program instance created for this program name.

[ENTRY_POINT]
is the address of the entry point of the last program instance created for this program name.

[LOCATION] indicates where the program for which the LOAD_POINT and ENTRY_POINT have been returned resides.

[ACCESS] is the type of storage that the program resides in.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	END_LIST
INVALID	INVALID_BROWSE_TOKEN

LDLD gate, GET_NEXT_INSTANCE function

The GET_NEXT_INSTANCE function of the LDLD gate is used to browse the current program instances in ascending load point address sequence.

Input parameters

BROWSE_TOKEN is a valid browse token as returned by the preceding START_BROWSE request.

Output parameters

[PROGRAM_NAME]
is the name of the program of which this is an instance.

[PROGRAM_TYPE]
is the current program copy type.

[PROGRAM_USAGE]
is the current usage definition.

[EXECUTION_KEY]
is the execution key for the program.

[PROGRAM_ATTRIBUTE]
is the current residency attribute of the program.

[SPECIFIED_AMODE]
is the current addressing mode required by CICS for the program. If REQUIRED_AMODE was omitted when the program was defined, AMODE_NOT_SPECIFIED is returned.

[SPECIFIED_RMODE]
is the current residency mode required by CICS for the program. If REQUIRED_RMODE was omitted when the program was defined, RMODE_NOT_SPECIFIED is returned.

[PROGRAM_LENGTH]
is the length of the program in bytes. If the program has not been used, this is zero.

[ENTRY_POINT]
is the address of the entry point of the last program instance created for this program name.

[LOAD_POINT] is the address of the load point of the last program instance created for this program name.

[LOCATION] indicates where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

[ACCESS] is the type of storage that the program resides in.

[INSTANCE_USE_COUNT]
is the current number of users of this instance.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	END_LIST
INVALID	INVALID_BROWSE_TOKEN

LDLD gate, END_BROWSE function

The END_BROWSE function of the LDLD gate is used to end a browse session.

Input parameters

BROWSE_TOKEN is the token identifying this browse session.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_BROWSE_TOKEN

LDLD gate, IDENTIFY_PROGRAM function

The IDENTIFY_PROGRAM function of the LDLD gate is used to locate the program instance which contains the specified address.

Input parameters

ADDRESS is a storage address.

Output parameters

[PROGRAM_NAME]
is the name of the program of which this is an instance.

[PROGRAM_TYPE]
is the current program copy type.

[PROGRAM_USAGE]
is the current usage definition.

[EXECUTION_KEY]
is the execution key for the program.

[PROGRAM_ATTRIBUTE]
is the current residency attribute of the program.

[SPECIFIED_AMODE]
is the addressing mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_AMODE was omitted when the program was defined, AMODE_NOT_SPECIFIED is returned.

[SPECIFIED_RMODE]
is the residency mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_RMODE was omitted when the program was defined, RMODE_NOT_SPECIFIED is returned.

[PROGRAM_LENGTH]
is the length of the program in bytes. If the program has not been used, this is zero.

[ENTRY_POINT]
is the address of the entry point of the last program instance created for this program name.

[LOAD_POINT] is the address of the load point of the last program instance created for this program name.

[LOCATION] indicates where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

[ACCESS] is the type of storage that the program resides in.

[INSTANCE_USE_COUNT]
is the current number of users of this instance.

[CSECT_NAME] is the name of the CSECT within the module which contains the address. If no CSECT is available, the module name is returned.

[OFFSET_INTO_CSECT]
is the offset of the address within the CSECT. If no CSECT is available, the module name is returned.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	INSTANCE_NOT_FOUND

LDLD gate, SET_OPTIONS function

The SET_OPTIONS function of the LDLD gate is used to set loader global options.

Input parameters

[LLACOPY]
indicates whether the loader is to use the MVS macro LLACOPY or BLDL to locate programs. It can have any of these values:

YES|NO|NEWCOPY

[SHARED_PROGRAMS]
indicates whether the loader is to use LPA-resident programs to satisfy ACQUIRE requests. It can have either of these values:

YES|NO

[STORAGE_FACTOR]
indicates the percentage of system free storage that may be occupied by program instances that have a zero use count.

[PRVMOD]
is a list of the names of modules that are not to be used from the MVS link pack area (LPA), but instead are to be loaded as private copies from the DFHRPL library.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	CATALOG_NOT_OPERATIONAL CATALOG_ERROR
INVALID	INVALID_STORAGE_FACTOR

LDLD gate, INQUIRE_OPTIONS function

The INQUIRE_OPTIONS function of the LDLD gate is used to return loader global options.

Input parameters

None.

Output parameters

[SHARED_PROGRAMS]
indicates whether the loader is utilizing LPA-resident programs to satisfy ACQUIRE requests.

[STORAGE_FACTOR]
indicates the percentage of system free storage that may be occupied by program instances that have a zero use count.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

LDLD gate, CATALOG_PROGRAMS function

The CATALOG_PROGRAMS function of the LDLD gate is used at the end of CICS initialization to request the loader domain to catalog all the program definitions that need cataloging. The call is issued by the DFHSIJ1 module.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	CATALOG_NOT_OPERATIONAL CATALOG_ERROR

Loader domain's generic gates

Table 62 summarizes the loader domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 62. Loader domain's generic gates

Gate	Trace	Function	Format
DMDM	LD 6001 LD 6002	PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	LD 5001 LD 5002	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
SMNT	LD 4001 LD 4002	STORAGE_NOTIFY	SMNT

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669
Format STST—"Statistics domain's generic format" on page 1198
Format SMNT—"Storage manager domain's generic formats" on page 1159

In preinitialization processing, the LDLD gate is added, enabling programs to be loaded.

Loader domain (LD)

In initialization processing, on a cold start, the loader domain purges the loader program definitions (for user application programs and non-nucleus CICS modules) from the CICS global catalog. The loader domain then reads program definitions from the local catalog, and makes them available to CICS.

On a warm or emergency start, the loader domain reads program definitions from the global and local CICS catalogs, and makes them available to CICS.

For any type of start, the loader domain loads the subset of CICS nucleus programs that are defined as resident.

In quiesce and termination processing, the loader domain performs only internal routines.

Modules

Module	Function
DFHLDDM	Handles the following requests: PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHLDDMI	Reinstates any program resources defined during previous runs of CICS. It is called by DFHLDDM.
DFHLDDUF	Formats the loader domain control blocks in a CICS system.
DFHLDLD	Directs the following requests to DFHLDLD1, DFHLDLD2, or DFHLDLD3, as appropriate: ACQUIRE_PROGRAM RELEASE_PROGRAM REFRESH_PROGRAM DEFINE_PROGRAM DELETE_PROGRAM INQUIRE_PROGRAM START_BROWSE GET_NEXT_PROGRAM GET_NEXT_INSTANCE END_BROWSE IDENTIFY_PROGRAM SET_OPTIONS INQUIRE_OPTIONS CATALOG_OPTIONS
DFHLDLD1	Handles the following requests: ACQUIRE_PROGRAM RELEASE_PROGRAM REFRESH_PROGRAM
DFHLDLD2	Handles the following requests: DEFINE_PROGRAM DELETE_PROGRAM
DFHLDLD3	Handles the following requests: INQUIRE_PROGRAM START_BROWSE GET_NEXT_PROGRAM GET_NEXT_INSTANCE END_BROWSE IDENTIFY_PROGRAM SET_OPTIONS INQUIRE_OPTIONS CATALOG_OPTIONS

Module	Function
DFHLDNT	Handles the following request: STORAGE_NOTIFY
DFHLDST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHLDSVC	Provides authorized services for loader domain functions that involve MVS load facilities.
DFHLDTRI	Provides a trace interpretation routine for CICS dumps and traces.

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the loader domain are of the form LD xxxx; the corresponding trace levels are LD 1, LD 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 88. Log manager domain (LG)

The log manager domain (also sometimes known simply as “log manager” or “logger”) provides facilities for Recovery Manager to:

- Write records to the CICS system log
- Read records from the CICS system log
- Maintain the system log deleting obsolete records and shunting old, but still needed, records to a secondary system log.

It also provides facilities to:

- Write user journal, forward recovery and auto journals records to MVS system logger logstreams or the MVS SMF log.
- Install, discard and inquire for Journalmodel resource definitions
- Auto-install, discard, inquire and set for Journal definitions
- Connect, disconnect and define for MVS system logger logstreams
- Collect statistics for Journal and Logstream usage.

Log manager domain’s specific gates

Table 63 summarizes the log manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 63. Log manager domain’s specific gate

Gate	Trace	Function	XPI
LGGL	LG 0201	OPEN	NO
	LG 0202	WRITE	NO
		FORCE	NO
		CLOSE	NO
		WRITE_JNL	NO
		FORCE_JNL	NO
		UOW_TIME	NO
		INITIALIZE	NO
LGJN	LG 0301	INQUIRE	NO
	LG 0302	START_BROWSE	NO
	LG 0314	GET_NEXT	NO
	LG 0325	END_BROWSE	NO
		SET	NO
		DISCARD	NO
		EXPLICIT_OPEN	NO
		IMPLICIT_OPEN	NO
		INITIALIZE	NO
		STREAM_FAIL	NO
		PROCESS_STATISTICS	NO
LGLD	LG 0401	INQUIRE	NO
	LG 0402	START_BROWSE	NO
	LG 0411	GET_NEXT	NO
	LG 0412	END_BROWSE	NO
	LG 0415	MATCH	NO
		INSTALL	NO
		DISCARD	NO
		INITIALIZE	NO
LGST	LG 0501	INQUIRE	NO
	LG 0502	START_BROWSE	NO
	LG 0514	GET_NEXT	NO
	LG 0517	END_BROWSE	NO
	LG 0526	CONNECT	NO
		DISCONNECT	NO
		INITIALIZE	NO
LGPA	LG 0601	INQUIRE_PARAMETERS	YES
	LG 0602	SET_PARAMETERS	YES

Table 63. Log manager domain's specific gate (continued)

Gate	Trace	Function	XPI
LGLB	LG 2001 LG 2002	CONNECT DISCONNECT GL_WRITE GL_FORCE DISCONNECT_ALL	NO NO NO NO NO
LGCC	LG 2101 LG 2102	SYSINI CREATE_CHAIN_TOKEN RELEASE_CHAIN_TOKEN RESTORE_CHAIN_TOKEN START_BROWSE_CHAINS BROWSE_CHAINS_GET_NEXT END_BROWSE_CHAINS DELETE_ALL SET_HISTORY DELETE_HISTORY SET_KEYPOINT_FREQUENCY INQUIRE_KEYPOINT_FREQUENCY SET_DEFER_INTERVAL INQUIRE_DEFER_INTERVAL INQUIRE_KEYPOINT_STATS RESET_KEYPOINT_STATS	NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO NO
LGWF	LG 2201 LG 2202	WRITE FORCE_DATA	NO NO
LGCB	LG 2301 LG 2302	START_CHAIN_BROWSE CHAIN_BROWSE_GET_NEXT END_CHAIN_BROWSE	NO NO NO
LGBA	LG 2401 LG 2402	START_BROWSE_ALL BROWSE_ALL_GET_NEXT END_BROWSE_ALL	NO NO NO
LGMV	LG 2501 LG 2502	MOVE_CHAIN	NO
LGSR	LG 2601 LG 2602	WRITE FORCE_DATA	NO NO

LGBA gate, BROWSE_ALL_GET_NEXT function

Returns the next record in the browse all object.

Input parameters

None

Output parameters

USER_TOKEN is a user token that was passed in by RESTORE_CHAIN_TOKEN.

USER_DATA is the address of the CICS record just read from the CICS system log.

USER_DATA_LENGTH
is the length of the CICS record just read from the chain.

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	END_OF_DATA

LGBA gate, END_BROWSE_ALL function

Destroys the browse all object.

Input parameters

None

Output parameters

RESPONSE
is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGBA gate, START_BROWSE_ALL function

Creates a browse all object for the CICS system log.

Input parameters

None

Output parameters

RESPONSE
is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGCB gate, CHAIN_BROWSE_GET_NEXT function

Creates a browse object for the chain denoted by CHAIN_TOKEN.

Input parameters

CHAIN_TOKEN
is a chain token.

Output parameters

USER_DATA
is the address of the CICS record just read from the chain.

USER_DATA_LENGTH
is the length of the CICS record just read from the chain.

RESPONSE
is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	END_OF_DATA

LGCB gate, END_CHAIN_BROWSE function

Destroys the chain browse object denoted by CHAIN_TOKEN.

Input parameters

CHAIN_TOKEN
is a chain token.

Output parameters

RESPONSE
is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGCB gate, START_CHAIN_BROWSE function

Creates a browse object for the chain denoted by CHAIN_TOKEN.

Input parameters

CHAIN_TOKEN
is a chain token.

Output parameters

RESPONSE
is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGCC gate, SYSINI function

Creates a primary and secondary log stream objects of type MVS that comprises the CICS system log.

Input parameters

None

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call.

LGCC gate, CREATE_CHAIN_TOKEN function

Creates a CHAIN TOKEN.

Input parameters

None

Output parameters

CHAIN_TOKEN is a new chain token token, which can be used as input to RELEASE_CHAIN_TOKEN, RESTORE_CHAIN_TOKEN, START_CHAIN_BROWSE, CHAIN_BROWSE_GET_NEXT, END_CHAIN_BROWSE, MOVE_CHAIN

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call.

LGCC gate, RELEASE_CHAIN_TOKEN function

Destroys the chain token in CHAIN_TOKEN

Input parameters

CHAIN_TOKEN is a chain token that must have been created by CREATE_CHAIN_TOKEN or RESTORE_CHAIN_TOKEN

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call.

LGCC gate, RESTORE_CHAIN_TOKEN function

Creates a chain token and adds the last record (viewed as a chain element) read from the system log during a BROWSE_ALL_GET_NEXT

Input parameters

USER_TOKEN is a user token that is returned by BROWSE_CHAINS_GET_NEXT and BROWSE_ALL_GET_NEXT.

Output parameters

CHAIN_TOKEN is a new chain token token, which can be used as input to RELEASE_CHAIN_TOKEN, RESTORE_CHAIN_TOKEN, START_CHAIN_BROWSE, CHAIN_BROWSE_GET_NEXT, END_CHAIN_BROWSE, MOVE_CHAIN.

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call.

LGCC gate, START_BROWSE_CHAINS function

Creates a chains browse object and initializes the browse cursor position.

Input parameters
None

Output parameters
RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call.

LGCC gate, BROWSE_CHAINS_GET_NEXT function
Returns the next chain token and moves the browse cursor position to the next chain.

Input parameters
None

Output parameters
CHAIN_TOKEN is the chain token of the next chain in the chains browse list.

USER_TOKEN is a user token that was passed in by RESTORE_CHAIN_TOKEN.

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	END_OF_CHAINS

LGCC gate, END_BROWSE_CHAINS function
Destroys the browse chains object.

Input parameters
None

Output parameters
RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call.

LGCC gate, DELETE_ALL function
Deletes all of the data on both log streams of the CICS system log.

Input parameters
None

Output parameters
RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call.

LGCC gate, SET_HISTORY function
Evaluates and saves the current history point for both log streams of the CICS system log. The history point of a log stream is the oldest block id that CICS knows of on the log stream.

Input parameters
None

Output parameters
RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call.

LGCC gate, DELETE_HISTORY function

Deletes all blocks of data, for both log streams of the CICS system log, that are older than the corresponding history point saved during a call of SET_HISTORY.

Input parameters

None

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call.

LGCC gate, SET_KEYPOINT_FREQUENCY function

Sets the activity frequency to KEYPOINT_FREQUENCY.

Input parameters

None

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	OUT_OF_RANGE

LGCC gate, INQUIRE_KEYPOINT_FREQUENCY function

Returns the activity keypoint frequency value in KEYPOINT_FREQUENCY.

Input parameters

None

Output parameters

KEYPOINT_FREQUENCY
is the current keypoint frequency value.

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call.

LGCC gate, SET_DEFER_INTERVAL function

Sets the log defer interval.

Input parameters

DEFER_INTERVAL
is the number of milliseconds for which a forced log write will be deferred. The maximum value that may be specified is 65535 milliseconds.

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	OUT_OF_RANGE

LGCC gate, INQUIRE_DEFER_INTERVAL function

Returns the number of millisecoonds for which a forced log write will be deferred.

Input parameters

None

Output parameters

DEFER_INTERVAL
is the number of millisecoonds for which a forced log write will be deferred.

RESPONSE
is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGCC gate, INQUIRE_KEYPOINT_STATS function

Return the number of keypoints that have occurred since the count was last reset.

Input parameters

None

Output parameters

KEYPOINT_COUNT
is the number of keypoints that have occurred since the count was last reset.

RESPONSE
is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGCC gate, RESET_KEYPOINT_STATS function

Reset the count of the number of keypoints.

Input parameters

None

Output parameters

RESPONSE
is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGGL gate, OPEN function

Opens a general log and returns a log token. The log token is used by the WRITE, FORCE and CLOSE operations.

Input parameters

STREAM_NAME
The 26-byte log stream name to be opened

JNL_NAME
The 8-byte journal name to be opened

Either STREAM_NAME or JNL_NAME must be specified

COMPONENT
Identifies the component (e.g. FC) opening this stream

[USER_TOKEN]
A token that identifies to the calling component why this log stream was opened. It will be passed to the ERROR gate in the event that an error is detected on the log stream. For example for file control it might contain a pointer to the DSNBx

[ERROR_GATE] The domain gate number that the logger should call using ERROR if an error occurs accessing the log stream.

Output parameters

LOG_TOKEN The token to be used on subsequent WRITE, FORCE, CLOSE requests.

LOG_TYPE The associated log stream type: It can have any one of these values:

- **MVS** MVS logger stream
- **SMF** SMF logging
- **DUMMY** No logging

JNL_STREAM The MVS logstream name associated with the journal being opened

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ERROR_OPENING_LOG LOG_IS_SYSTEM_LOG LOG_IS_DISABLED LOG_NOT_DEFINED LOG_HAS_FAILED INVALID_JNL_NAME
INVALID	INVALID_PARAMETERS

LGGL gate, WRITE function

Write a record to a general log identified by a token from a previous OPEN.

Input parameters

LOG_TOKEN The token returned by OPEN

DATA The address of a reusable Iliffe vector describing the items of data to be written to the log stream.

[FORCE_NOW] Indicates that the caller wishes to wait until the data has been successfully written to the log stream. It can have either of these values:

YES|NO

Default is NO

[FORCE_AT_SYNC]
Indicates that the caller wishes the log stream to be forced when the associated transaction reaches Syncpoint. It can have either of these values:

YES|NO

Default is NO

Note: Force_at_Sync can be used in conjunction with FORCE_NOW. This is needed by File control for ESDS writes which have to be forced immediately but which also need the UOW structure to allow the calculation of Fuzzy backup recovery times.

Output parameters

[FORCE_TOKEN]
A token to be used on a subsequent FORCE to ensure that a specific records and any prior records have been hardened

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	WRITE_ERROR BUFFER_LENGTH_ERROR
INVALID	UNKNOWN_LOG_TOKEN

LGGL gate, FORCE function

Ensures that the previously written records have been flushed from the buffer and hardened on the chosen log stream

Input parameters

LOG_TOKEN The token returned by OPEN

[FORCE_TOKEN]
Token returned by WRITE to indicate a specific record to be written. If omitted all records are forced.

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	WRITE_ERROR
INVALID	UNKNOWN_LOG_TOKEN

LGGL gate, CLOSE function

Invalidates the LOG_TOKEN, on the last usage of a log stream disconnects from the log stream

Input parameters

LOG_TOKEN The token returned by OPEN

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	WRITE_ERROR
INVALID	UNKNOWN_LOG_TOKEN

LGGL gate, WRITE_JNL function

Write a record to a general log identified by a journal name

Input parameters

JNL_NAME The 8-byte journal name to be written to

DATA The address of a reusable Iliffe vector describing the items of data to be written to the log stream.

[FORCE_NOW] Indicates that the caller wishes to wait until the data has been successfully written to the log stream. It can have either of these values:
YES|NO

Default is NO

[FORCE_AT_SYNC]
Indicates that the caller wishes the log stream to be forced when the associated transaction reaches Syncpoint. It can have either of these values:

YES|NO

Default is NO

Note: Force_at_Sync can be used in conjunction with FORCE_NOW. This is needed by File control for ESDS writes which have to be forced immediately but which also need the UOW structure to allow the calculation of Fuzzy backup recovery times.

COMPONENT
Identifies the component (e.g. TC) writing this stream

SUSPEND
Supported for compatibility with old EXEC interface. Causes BUFFER_FULL exception to be raised if there is no space rather than waiting for space. The task may still be suspended for many other reasons! It can have either of these values:

YES|NO

Default is YES

Output parameters

[FORCE_TOKEN]
A token to be used on a subsequent FORCE_JNL to ensure that a specific record and any prior records have been hardened

RESPONSE
is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	WRITE_ERROR ERROR_OPENING_LOG LOG_IS_SYSTEM_LOG LOG_IS_DISABLED LOG_HAS_FAILED LOG_NOT_DEFINED BUFFER_FULL INVALID_JNL_NAME BUFFER_LENGTH_ERROR

LGGL gate, FORCE_JNL function

Ensures that the previously written records have been hardened on the chosen log.

Input parameters

JNL_NAME
The 8-byte journal name to be forced

[FORCE_TOKEN]
Token returned by WRITE_JNL to indicate a specific record to be written. If omitted all records are forced.

Output parameters

RESPONSE
is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	WRITE_ERROR LOG_IS_NOT_ACTIVE LOG_IS_SYSTEM_LOG LOG_IS_DISABLED LOG_HAS_FAILED

LGGL gate, UOW_TIME function

Returns the oldest active transactions first log write time for use in calculating the recovery time for Backup while open.

Usually called by AKP processing when calculating the recovery time for non-RLS BWO files

Input parameters

UOW_TIME_STAMP
: The 8-byte STCK format time of the oldest active transaction that has written log records with the FORCE_AT_SYNC option, or current time if there are no active transactions.

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call

LGGL gate, INITIALIZE function

Establish subpools, locks, and anchor control blocks

Called as subroutine during domain initialization.

Input parameters

None

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call

LGJN gate, INQUIRE function

Returns information about the current state of a user journal

Also causes the stats information for a particular journal to be updated, when called as part of a FORCE_JNL request from LGGL.

Input parameters

JNL_NAME The 8-byte Journal name to be inquired upon

[FORCE] Indicates that a force of the data in the buffer has been requested.

This is used to indicate when the stats field in the journal info, which records the number of flushes, needs incrementing.

Output parameters

[LOG_TYPE] The associated log stream type:

- **MVS** MVS logger stream
- **SMF** SMF logging
- **DUMMY** No logging

[JNL_STATUS] The associated log stream status:

Note: Status will always appear as disconnected for journals that have not been used as user journals (i.e. system logs, forward recovery logs, fc auto journals) even though they may be in use

CONNECTED Stream is currently connected
DISCONNECTED Stream is not currently connected
DISABLED Stream has been disabled by SPI/CEMT function
FAILED The MVS log stream has failed

[STREAM_NAME]
The associated MVS log stream name. Blank for SMF or DUMMY

[SYSTEM_LOG] Whether or not the journal is a system log. It can have either of these values:

YES|NO

[STREAM_TOKEN]
The log stream token if the journal is currently connected to an MVS log stream or the logbuf token for an SMF journal.

If specified the stream shared lock will be acquired and it its the callers responsibility to free the lock when they have finished with the stream token.

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_JNL_NAME

LGJN gate, START_BROWSE function

Initialize browse token for subsequent GET_NEXT requests

Input parameters

None.

Output parameters

BROWSE_TOKEN Token for use on subsequent GET_NEXT requests

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None defined for this call

LGJN gate, GET_NEXT function

Return information for next Journal.

Input parameters

BROWSE_TOKEN Token returned by START_BROWSE

Output parameters

JNL_NAME The next 8-byte Journal name found

[LOG_TYPE] The associated log stream type:

MVS MVS logger stream
SMF SMF logging
DUMMY No logging

[JNL_STATUS] The associated log stream status:

Note: Status will always appear as disconnected for journals that have not been used as user journals (i.e. system logs, forward recovery logs, fc auto journals) even though they may be in use

CONNECTED Stream is currently connected
DISCONNECTED Stream is not currently connected

DISABLED Stream has been disabled by SPI/CEMT function
FAILED The MVS log stream has failed

[STREAM_NAME]
The associated MVS log stream name. Blank for SMF or DUMMY

[SYSTEM_LOG] Whether or not the journal is a system log. It can have either of these values:
YES|NO

RESPONSE is the log manager domain's response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_MORE_DATA_AVAILABLE
INVALID	INVALID_BROWSE_TOKEN

LGJN gate, END_BROWSE function

Terminate browse and invalidate browse token

Input parameters

BROWSE_TOKEN Token returned by START_BROWSE

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_BROWSE_TOKEN

LGJN gate, SET function

Update the status of the Journal.

Will create journal if it does not currently exist (except for FLUSH)

Input parameters

JNL_NAME The 8-byte Journal name to be updated

JNL_STATUS The new status for the journal:
CONNECTED Stream is to be connected
DISCONNECTED Stream is to be disconnected
DISABLED Stream is to be disabled from further use
FLUSH The current log buffers are to be written to the log stream

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	JNL_ALREADY_IN_REQ_STATE JNL_IS_NOT_ACTIVE LOG_IS_SYSTEM_LOG SYSTEM_LOG_CONFLICT UNKNOWN_JNL_NAME UNABLE_TO_CREATE_JNL ERROR_OPENING_LOG JNL_HAS_FAILED INVALID_JNL_NAME WRITE_ERROR

LGJN gate, DISCARD function

Remove a journal from the set of known journals to clean up the catalog or to allow it to be reinstalled with a new set of attributes.

Input parameters

JNL_NAME The 8-byte Journal name to be discarded

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LOG_IS_SYSTEM_LOG UNKNOWN_JNL_NAME

LGJN gate, EXPLICIT_OPEN function

Inquire on a journal and if the journal does not already exist in the set of known journals perform the autoinstall process to define it.

The stream is explicitly opened for each call and so must eventually be explicitly closed using the LGST DISCONNECT function

Input parameters

JNL_NAME The 8-byte Journal name to be Explicit_Opened

SYSTEM_LOG Whether or not this journal is to be used as a system log It can have either of these values:

YES|NO

Output parameters

[LOG_TYPE] The associated log stream type:

- **MVS** MVS logger stream
- **SMF** SMF logging
- **DUMMY** No logging

[JNL_STATUS] The associated log stream status:

Note: Status will always appear as disconnected for journals that have not been used as user journals (i.e. system logs, forward recovery logs, fc auto journals) even though they may be in use

- **CONNECTED** Stream is currently connected
- **DISCONNECTED** Stream is not currently connected
- **DISABLED** Stream has been disabled by SPI/CEMT function

FAILED The MVS log stream has failed

STREAM_TOKEN The log stream token if the journal is currently connected to an MVS log stream or the logbuf token for an SMF journal.

[STREAM_NAME]
The associated MVS log stream name. Blank for SMF or DUMMY

[LOG_TOKEN] The buffer manager's log token for the log stream

[STRUCTURE_NAME]
is the 16 byte name of the coupling facility structure of the log stream.

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNABLE_TO_CREATE_JNL SYSTEM_LOG_CONFLICT JNL_IS_DISABLED JNL_HAS_FAILED ERROR_OPENING_LOG INVALID_JNL_NAME

LGJN gate, IMPLICIT_OPEN function

Inquire on a journal and if the journal does not already exist in the set of known journals perform the autoinstall process to define it. If the associated log stream has not been opened then it is opened and the stream token returned.

Input parameters

JNL_NAME The 8-byte Journal name to be Explicit_Opened

SYSTEM_LOG Whether or not this journal is to be used as a system log It can have either of these values:

YES|NO

[FORCE] Indicates that a force of the data in the buffer has been requested.

This is used to indicate when the stats field in the journal info, which records the number of flushes, needs incrementing. It can have either of these values:

YES|NO

[WRITE_BYTES]
The number of bytes of data being written, as a 64 bit value.

This field is used to update the bytes counter in the stats information for a journal, and also indicates that the writes counter also needs incrementing.

Output parameters

[LOG_TYPE] The associated log stream type:

MVS MVS logger stream
SMF SMF logging
DUMMY No logging

[JNL_STATUS] The associated log stream status:

Note: Status will always appear as disconnected for journals that have not been used as user journals (i.e. system logs, forward recovery logs, fc auto journals) even though they may be in use

CONNECTED Stream is currently connected
DISCONNECTED Stream is not currently connected
DISABLED Stream has been disabled by SPI/CEMT function
FAILED The MVS log stream has failed

[STREAM_NAME]
The associated MVS log stream name. Blank for SMF or DUMMY

STREAM_TOKEN The log stream token if the journal is currently connected to an MVS log stream or logbuf token for SMF.

If specified the stream shared lock will be acquired and it its the callers responsibility to free the lock when they have finished with the stream token.

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNABLE_TO_CREATE_JNL SYSTEM_LOG_CONFLICT JNL_IS_DISABLED JNL_HAS_FAILED ERROR_OPENING_LOG INVALID_JNL_NAME

LGJN gate, INITIALIZE function

Establish subpools, locks, and anchor control blocks

Called as subroutine during domain initialization.

Input parameters

None

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None defined for this call

LGJN gate, STREAM_FAIL function

Marks all journals that have used the failing log stream as failed, issues a message, and closes the stream connection. This ensures that all subsequent activity for the log stream is rejected until either CICS is restarted or the operator explicitly reactivates the journal

Input parameters

STREAM_TOKEN The token of the log stream that has failed

STREAM_NAME The name of the log stream that has failed

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None defined for this call

LGJN gate, PROCESS_STATISTICS function

Deal with the various types of requests for journal statistics using the information in the STST parameter list.

Input parameters

STATS_PARMS The address of the STST parameter list.

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_JNL_NAME NO_JOURNALS_DEFINED

LGLB gate, CONNECT function

Creates a log stream object and if of type MVS, a connection is made to the log stream, denoted by its name, through the MVS logger.

Input parameters

STREAM_NAME is the name of the log stream to be connected. Only valid if the log type is MVS.

SYSTEM_LOG is the system log indicator, which can assume the following values:
- **YES** The log stream being connected is part of the CICS system log.
- **NO** The log stream being connected is general log.

LOG_TYPE is the log stream type, which can assume the following values:
- **MVS** A MVS logger log stream
- **SMF** The MVS SMF log
- **DUMMY** A dummy log

JOURNAL_NAME is the journal name associated with the log stream on this request.

[STRUCTURE_NAME]
is the 16 byte name of the coupling facility structure of the log stream.

Output parameters

LOGBUF_TOKEN is the token denoting the connected log stream, which can be used as input to GL_WRITE, GL_FORCE and DISCONNECT.

RESPONSE is the response code, possible values are:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LOG_NOT_DEFINED CONNECT_FAILURE

LGLB gate, DISCONNECT function

Destroys the log stream object and if it is of type MVS, disconnects from the MVS logger.

Input parameters

LOGBUF_TOKEN is the token of the log stream created during a call of CONNECT.

Output parameters

RESPONSE is the response code, possible values are:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LOG_NOT_DEFINED CONNECT_FAILURE

LGLB gate, GL_WRITE function

Writes a record to a general log denoted by LOGBUF_TOKEN.

Input parameters

LOGBUF_TOKEN is the token of the log stream created during a call of CONNECT.
DATA is the address of the data to be written.
COMPONENT identifies the original CICS component making this request.
SUSPEND is a task suspend indicator, which can assume the following values:
- **YES** The task may be suspended if necessary.
- **NO** If there is no buffer space immediately available without suspending the current task then return an exception with a reason of BUFFER_FULL

JOURNAL_NAME is the journal name associated with the log stream on this request.

Output parameters

FORCE_TOKEN is the token denoting the output buffer which includes the data of this request. This token can be used as input to GL_FORCE.
RESPONSE is the response code, possible values are:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BUFFER_FULL BUFFER_LENGTH_ERROR WRITE_FAILURE

LGLB gate, GL_FORCE function

Ensures that the output buffer denoted by FORCE_TOKEN for the log stream denoted by LOGBUF_TOKEN has been written to the physical media.

Input parameters

LOGBUF_TOKEN is the token of the log stream created during a call of CONNECT.
FORCE_TOKEN is the token denoting the output buffer containing the data written during a call of GL_WRITE. A null token denotes the current output buffer.

Output parameters

RESPONSE is the response code, possible values are:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	WRITE_FAILURE

LGLB gate, DISCONNECT_ALL function

Ensures that any data in the output buffer has been written to the physical media before the stream connection is destroyed for all connected streams.

Input parameters

None.

Output parameters

RESPONSE is the response code, possible values are:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] No reason codes are defined for this call

LGLD gate, INQUIRE function

Returns information about the current state of a JournalModel

Input parameters

JOURNALMODEL_NAME
The 8-byte JournalModel name to be inquired upon

Output parameters

[JNL_TEMPLATE]
The associated journal name template

[LOG_TYPE] The associated log stream type:
- **MVS** MVS logger stream
- **SMF** SMF logging
- **DUMMY** No logging

[STREAM_PROTOTYPE]
The associated MVS log stream name prototype

RESPONSE is the log manager domain's response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_JOURNALMODEL_NAME

LGLD gate, START_BROWSE function

Initialize browse token for subsequent GET_NEXT requests

Input parameters

None

Output parameters

BROWSE_TOKEN Token for use on subsequent GET_NEXT requests

RESPONSE is the log manager domain's response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None defined for this function.

LGLD gate, GET_NEXT function

Return information for next JournalModel entry

Input parameters

BROWSE_TOKEN Token returned by START_BROWSE

Output parameters

JOURNALMODEL_NAME
The next 8-byte JournalModel name

[JNL_TEMPLATE]
The associated journal name template

[LOG_TYPE] The associated log stream type:
- **MVS** MVS logger stream
- **SMF** SMF logging
- **DUMMY** No logging

[STREAM_PROTOTYPE]
The associated MVS log stream name prototype

RESPONSE is the log manager domain's response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_MORE_DATA_AVAILABLE
INVALID	INVALID_BROWSE_TOKEN

LGLD gate, END_BROWSE function

Terminate browse and invalidate browse token

Input parameters

BROWSE_TOKEN Token returned by START_BROWSE

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_BROWSE_TOKEN

LGLD gate, MATCH function

Find JournalModel entry that best matches a journal name. Variables in the stream name prototype are resolved and the resultant stream name is returned.

Input parameters

JNL_NAME The journal name to be matched

Output parameters

LOG_TYPE The associated log stream type:
- **MVS** MVS logger stream
- **SMF** SMF logging
- **DUMMY** No logging

STREAM_NAME The MVS log stream name

RESPONSE is the log manager domain's response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_JNL_NAME

LGLD gate, INSTALL function

Create/replace JournalModel entry

Input parameters

JOURNALMODEL_NAME
The 8-byte JournalModel name

JNL_TEMPLATE The associated journal name template

LOG_TYPE The associated log stream type:
- **MVS** MVS logger stream
- **SMF** SMF logging
- **DUMMY** No logging

STREAM_PROTOTYPE
The associated MVS log stream name prototype

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_JNL_TEMPLATE INVALID_STREAM_PROTOTYPE

LGLD gate, DISCARD function

Remove a JournalModel from the set of defined JournalModels

Input parameters

JOURNALMODEL_NAME
The 8-byte JournalModel name to be discarded

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_JOURNALMODEL_NAME

LGLD gate, INITIALIZE function

Establish subpools, locks, and anchor control blocks

Called as subroutine during domain initialization.

Input parameters

None

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None defined for this function.

LGMV gate, MOVE_CHAIN function

Destroys the chain browse object denoted by CHAIN_TOKEN.

Input parameters

CHAIN_TOKEN is a chain token denoting the chain to be moved.

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None defined for this function.

LGPA gate, INQUIRE_PARAMETERS function

Inquire logger domain parameters.

Input parameters

None

Output parameters

[KEYPOINT_FREQUENCY]
: How often, in terms of physical writes to the system log, activity keypoints are initiated. A value of zero indicates that activity keypoints are not initiated.

RESPONSE
: is the log manager domain's response to the call. It can have any one of these values:

  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
: None defined for this function.

LGPA gate, SET_PARAMETERS function

Set logger domain parameters.

Input parameters

[KEYPOINT_FREQUENCY]
: How often, in terms of physical writes to the system log, activity keypoints should be initiated. A value of zero indicates that activity keypoints should not be initiated.

  Non-zero values outside the range from 200 to 65535 inclusive are invalid and cause the OUT_OF_RANGE exception to be returned.

Output parameters

RESPONSE
: is the log manager domain's response to the call. It can have any one of these values:

  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
: is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	OUT_OF_RANGE

LGSR gate, LOGSTREAM_STATS function

Collects, and resets if required, the log stream statistics of either the log stream denoted by LOGSTREAM_NAME or of all log streams known to the log manager.

Input parameters

[ALL]
: if specified then the request is for all log streams of type MVS known to the log manager.

[LOGSTREAM_NAME]
: if specified then this is a log stream name, which must be of type MVS.

STATS_BUFFER_ADDR
: is the address of a buffer to put the log stream statistics record(s).

STATS_BUFFER_LENGTH
: is the length of the buffer.

[RESET]
: is a request qualifier that assumes the following values:

  YES The log stream statistics data are to be reset after collection.

  NO The log stream statistics data are not to be reset.

Output parameters

RESPONSE
: is the log manager domain's response to the call. It can have any one of these values:

  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
: is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LOG_NOT_DEFINED

LGST gate, INQUIRE function

Returns information about the current state of a stream name

Input parameters

STREAM_NAME The 26-byte stream name

Output parameters

[USE_CT] The current number of users of the stream

[SYSTEM_LOG] Whether or not this is a CICS system log It can have either of these values:

YES|NO

[FAILED] Whether or not the stream has failed It can have either of these values:

YES|NO

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_STREAM_NAME

LGST gate, START_BROWSE function

Initialize browse token for subsequent GET_NEXT requests

Input parameters

None

Output parameters

BROWSE_TOKEN Token for use on subsequent GET_NEXT requests

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None defined for this function.

LGST gate, GET_NEXT function

Return information for next stream entry

Input parameters

BROWSE_TOKEN Token returned by START_BROWSE

Output parameters

STREAM_NAME The 26-byte stream name

[USE_CT] The current number of users of the stream

[SYSTEM_LOG] Whether or not this is a CICS system log It can have either of these values:

YES|NO

[FAILED] Whether or not the stream has failed It can have either of these values:

YES|NO

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_MORE_DATA_AVAILABLE
INVALID	INVALID_BROWSE_TOKEN

LGST gate, END_BROWSE function

Terminate browse and invalidate browse token

Input parameters

BROWSE_TOKEN Token returned by START_BROWSE

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_BROWSE_TOKEN

LGST gate, CONNECT function

Connect to an MVS log stream, or increment use count on subsequent call.

Input parameters

STREAM_NAME The 26-byte stream name

SYSTEM_LOG Whether or not this is a CICS system log It can have either of these values:

YES|NO

Output parameters

STREAM_TOKEN A token to represent this stream

[STRUCTURE_NAME]
is the 16 byte name of the coupling facility structure of the log stream.

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SYSTEM_LOG_CONFLICT LOG_HAS_FAILED DEFINE_FAILURE CONNECT_FAILURE,

LGST gate, DISCONNECT function

Decrement the stream use count and disconnect from the MVS logger on last use

Input parameters

STREAM_TOKEN The token returned by CONNECT

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None defined for this function.

LGST gate, INITIALIZE function

Establish subpools, locks, and anchor control blocks

Called as subroutine during domain initialization.

Input parameters

None

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None defined for this function.

LGWF gate, FORCE_DATA function

Ensures that the output buffer denoted by FORCE_TOKEN has been written to the physical media.

Input parameters

FORCE_TOKEN is a token denoting the output buffer containing the data written during a call of GL_WRITE. A null token denotes the current output buffer.

Output parameters

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	AKP_KICK_OFF

LGWF gate, WRITE function

Writes a record to the CICS system log.

Input parameters

DATA is the address of the data to be written.

CHAIN_TOKEN is a token denoting the chain that this record belongs. A chain token is created by CREATE_CHAIN_TOKEN and RESTORE__CHAIN_TOKEN

SUSPEND is a task suspend indicator, which can assume the following values:

- **YES** The task may be suspended if necessary.
- **NO** If there is no buffer space immediately available without suspending the current task then return an exception with a reason of BUFFER_FULL

FORCE is a request qualifier, which can assume the following values:

- **YES** The data of this request including any other data already in the output buffer is to be written to the physical media before returning.
- **NO** The data of this request need only be written to the output buffer, but may get written to the physical media.

RAISE_LENGERR

is a request qualifier, which can assume the following values:

- **YES** If the data length is too large to fit into the output buffer then an EXCEPTION condition is returned to the caller.
- **NO** If the data length is too large to fit into the output buffer then the log manager terminates CICS.

Output parameters

FORCE_TOKEN is the token denoting the output buffer which includes the data of the request. This token can be used as input to GL_FORCE.

RESPONSE is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BUFFER_FULL AKP_KICK_OFF BUFFER_LENGTH_ERROR

Log manager domain's generic gates

Table 64 summarizes the log manager domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 64. Log manager domain's generic gate

Gate	Trace	Function	Format
APUE	LG 0101 LG 0102	SET_EXIT_STATUS	APUE
DMDM	LG 0101 LG 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	LG 0701 LG 0702	COLLECT_STATISTICS COLLECT_RESOURCE_STATISTICS	STST

You can find descriptions of these functions and their input and output parameters, in the section. dealing with the corresponding generic format, in "Domain manager domain's generic formats" on page 669.

In Initialization processing, the log manager domain retrieves Journal and Journalmodel information from the catalog and initializes the system log except on a cold start when system log initialization occurs after group list install has completed.

In Quiesce processing, the log manager disconnects from MVS log streams after all transactions have completed.

Log manager domain's call back gates

Table 64 summarizes the log manager domain's call back gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the format for calls to the gate.

Table 65. Log manager domain's call back gate

Gate	Trace	Function	Format
RMRO	LG 0201 LG 0202	PERFORM_COMMIT PERFORM_PREPARE START_BACKOUT DELIVER_BACKOUT_DATA END_BACKOUT PERFORM_SHUNT PERFORM_UNSHUNT	RMRO

You can find descriptions of these functions and their input and output parameters, in the section. dealing with the recovery manager formats, in Chapter 99, "Recovery Manager Domain (RM)," on page 1061.

For PERFORM_PREPARE, PERFORM_COMMIT, END_BACKOUT the log manager forces any log buffers written using the FORCE_AT_SYNCH option of the LGGL WRITE gate to the MVS system logger. For the other RMRO gate functions the log manager does nothing.

Log manager domain's call back format

Table 66 describes the call back format owned by the log manager domain and shows the function performed on the calls.

Table 66. Call back format owned by the log manager domain

Format	Calling module	Function
LGGL	DFHLGGL	ERROR

In the descriptions of the formats that follow, the "input" parameters are input not to log manager domain, but to the domain being called by the log manager. Similarly, the "output" parameters are output by the domain that was called by log manager domain, in response to the call.

LGGL gate, ERROR function

This is a back-to-front or outbound function. The logger will call the domain that issued OPEN, using the gate number specified in ERROR_GATE, when a long term error condition is detected on the opened log stream.

The called domain should take any recovery action needed and close the log stream (if appropriate).

Called by the logger during log stream error processing.

Note: An error call back could occur while an Open or Close request for the associated log-token is still in progress.

Input parameters

ERROR_TYPE
Indicates the severity of the error: It can have either of these values:

LONG_TERM|RECOVERED

STREAM_NAME
The 26-byte name of the failing log stream name

[JNL_NAME]
The 8-byte journal name if the open was by journal name

COMPONENT
The 2-byte component id supplied on OPEN

USER_TOKEN
The 8-byte token supplied on OPEN, this allows the opening domain to determine what resource (eg DSNB) this open is associated with.

LOG_TOKEN
The token returned by OPEN

Output parameters

RESPONSE
is the log manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call

Modules

Module	Function
DFHLGDM	Log manager domain initialization and termination. Also handles exit activation for XLGSTRM and XRSINDI. Handles the DMDM and APUE gate functions
DFHLGDUF	A routine to format system dump information
DFHLGGL	Handles the LGGL and RMRO gate functions
DFHLGHB	Assesses the availability of the MVS system logger
DFHLGICV	Log record conversion for SSI exit
DFHLGIGT	Log record get routine for SSI exit
DFHLGILA	Lexical analysis for SSI exit
DFHLGIMS	Message composer for SSI exit
DFHLGIPA	Parser for SSI exit
DFHLGIPI	Parse interface for SSI exit
DFHLGISM	Parse message exit for SSI exit
DFHLGJN	Handles the LGJN gate functions
DFHLGLD	Handles the LGLD gate functions
DFHLGPA	Handles the LGPA gate functions

Log manager domain (LG)

Module	Function
DFHLGSC	Handles the STST gate functions
DFHLGST	Handles the LGST gate functions
DFHLGSSI	Handles the batch QSAM access to CICS logstreams via the DD SUBSYS=(LOGR...) SSI interface
DFHLGTRI	A routine to format trace points
DFHL2DM	Initializes the 'L2' part of the Log Manager Domain
DFHL2TRI	A routine to format the 'L2' trace points
DFHL2LB	Handles the LGLB gate functions
DFHL2SR	Handles the LGSR gate functions
DFHL2WF	Handles the LGWF gate functions
DFHL2CC	Handles the LGCC gate functions
DFHL2CB	Handles the LGCB gate functions
DFHL2BA	Handles the LGBA gate functions
DFHL2MV	Handles the LGMV gate functions
DFHL2BL1	Initializes the Block class data
DFHL2BL2	Retrieves the current block on the CICS system log
DFHL2BS1	Initializes the BrowseableStream class data
DFHL2BS2	Creates a BrowseableStream class instance
DFHL2BS3	Destroys a BrowseableStream class instance
DFHL2BS4	Destroys all BrowseableStream class instance
DFHL2CH1	Initializes the Chain class data
DFHL2CH2	Creates a Chain class instance
DFHL2CH3	Handles start chain browse
DFHL2CH4	Handles chain browse get next
DFHL2CH5	Handles end chain browse
DFHL2CHA	Handles start browse all
DFHL2CHN	Handles browse all get next
DFHL2CHL	Handles end browse all
DFHL2CHH	Handles start browse chains
DFHL2CHG	Handles browse chains get next
DFHL2CHI	Handles end browse chains
DFHL2CHR	Handles chain restore
DFHL2CHS	handles set history point
DFHL2CHE	Handles delete at history point
DFHL2CHM	Handles move chain
DFHL2HS2	Handles the log stream connect request to the MVS logger
DFHL2HS3	Handles the log stream disconnect request to the MVS logger
DFHL2HS4	Handles the log stream delete all request to the MVS logger
DFHL2HS5	Handles the log stream delete history request to the MVS logger
DFHL2HS6	Handles the log stream start browse block request to the MVS logger
DFHL2HS7	Handles the log stream start browse cursor request to the MVS logger

Module	Function
DFHL2HS8	Handles the log stream read browse cursor request to the MVS logger
DFHL2HS9	Handles the log stream end browse cursor request to the MVS logger
DFHL2HSG	Handles the log stream read browse block request to the MVS logger
DFHL2HSJ	Handles the log stream end browse block request to the MVS logger
DFHL2OFI	Initializes the ObjectFactory instance data
DFHL2SL1	Initializes the SystemLog class data
DFHL2SLN	Handles system log log stream open request
DFHL2SLE	Handles system log log stream failure notification
DFHL2SR1	Initializes the Stream class data
DFHL2SR2	Creates a Stream class instance
DFHL2SR3	Destroys a Stream class instance
DFHL2SR4	Collect and resets Stream statistics
DFHL2SR5	Destroys all Stream class instances
DFHL2VPX	Initializes the VariablePool class data

Exits

Two global user exit points are provided in this domain.

XLGSTRM
This exit is called prior to defining a new log stream to the MVS system logger

XRSINDI
This exit is called when a Journal or Journalmodel is installed or discarded. It is also called when CICS connects or disconnects an MVS system logger logstream.

See *CICS Customization Guide* for further information.

Trace

The point IDs for the log manager domain are of the form LG xxxx; the corresponding trace levels are LG 1, LG 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 89. Lock manager domain (LM)

The lock manager domain (also sometimes known simply as "lock manager") provides both locking and associated queuing facilities for CICS resources. Before using these facilities, a resource must add a named lock for itself. This lock can then be requested as either exclusive or shared. If an exclusive lock is obtained, no other task may obtain the lock with that name; if a shared lock is obtained, multiple tasks may obtain that lock, and the exclusive lock with that name cannot be acquired.

Lock manager domain's specific gate

Table 67 summarizes the lock manager domain's specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 67. Lock manager domain's specific gate

Gate	Trace	Function	XPI
LMLM	LM 0003	ADD_LOCK	NO
	LM 0004	DELETE_LOCK	NO
		LOCK	NO
		UNLOCK	NO
		TEST_LOCK_OWNER	NO

LMLM gate, ADD_LOCK function

The ADD_LOCK function of the LMLM gate is used to add a named lock to LM's state.

Input parameters

LOCK_NAME is an 8-character name.

Output parameters

LOCK_TOKEN is the 8-character token that uniquely identifies the lock, returned to the caller on the this call.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|INVALID|DISASTER|KERNERROR

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	INSUFFICIENT_STORAGE ABEND LOOP

LMLM gate, LOCK function

The LOCK function of the LMLM gate is used to request the lock.

Input parameters

LOCK_TOKEN is the token returned to the caller on the ADD_LOCK call.

MODE defines the type of lock. It can have either of these values:

EXCLUSIVE|SHARED

[WAIT] indicates whether a task is suspended (CICS) or a LOCK_BUSY is to be returned as a reason output parameter (NO). It can have either of these values:

CICS|NO

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|PURGED|INVALID|DISASTER|KERNERROR

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	INSUFFICIENT_STORAGE ABEND LOOP
EXCEPTION	LOCK_TOKEN_NOT_FOUND DUPLICATE_LOCK_OWNER LOCK_BUSY
Note: DUPLICATE_LOCK_OWNER is returned when a resource requests a lock twice without unlocking during the same task: this is often treated in the same way as OK by the requesting resource.	

LMLM gate, UNLOCK function

The UNLOCK function of the LMLM gate is used to release the lock.

Input parameters

LOCK_TOKEN is the token returned to the caller on the ADD_LOCK call.

MODE defines the type of lock to be released. It can have either of these values:

EXCLUSIVE|SHARED

[OWNER_TOKEN]
defines the owner of the lock.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	LOCK_TOKEN_NOT_FOUND SHARED_LOCK_FREE NOT_LOCK_OWNER

LMLM gate, TEST_LOCK_OWNER function

The TEST_LOCK_OWNER function of the LMLM gate is used to test the owner of a lock for self.

Input parameters

LOCK_TOKEN is the token returned to the caller on the ADD_LOCK call.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LOCK_TOKEN_NOT_FOUND NOT_LOCK_OWNER
DISASTER	ABEND LOOP

LMLM gate, DELETE_LOCK function

The DELETE_LOCK function of the LMLM gate is used to delete the named lock from LM's state.

Input parameters

LOCK_TOKEN
is the token returned to the caller on the ADD_LOCK call.

[OWNER_TOKEN]
defines the owner of the lock.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	LOCK_TOKEN_NOT_FOUND NOT_LOCK_OWNER

Lock manager domain's generic gates

Table 68 summarizes the lock manager domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 68. Lock manager domain's generic gates

Gate	Trace	Function	Format
DMDM	LM 0001 LM 0002	PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
DSNT	LM 0005 LM 0006	DISPATCHER_NOTIFY	DSNT

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669
Format DSNT—"Dispatcher domain's generic formats" on page 717

In preinitialization processing, gates are added to make lock manager services available to other domains.

In initialization, quiesce, and termination processing, the lock manager domain performs only internal routines.

Modules

Module	Function
DFHLMDM	Handles the following requests: PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHLMDS	Handles transaction manager domain MXT_CHANGE_NOTIFY requests.
DFHLMDUF	Formats the LM domain control blocks
DFHLMLM	Handles the following requests: ADD_LOCK DELETE_LOCK LOCK TEST_LOCK_OWNER UNLOCK
DFHLMTRI	Interprets LM domain trace entries

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the lock manager domain are of the form LM xxxx; the corresponding trace levels are LM 1, LM 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 90. Message domain (ME)

The message domain acts as a repository for CICS messages, and handles the sending of messages to transient data destinations or to the console. It also provides an interface for returning the text of a message to the caller.

Message domain's specific gates

Table 69 summarizes the message domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 69. Message domain's specific gates

Gate	Trace	Function	XPI
MEBM	None	INQUIRE_MESSAGE_DEFINITION INQUIRE_MESSAGE_LENGTH RETRIEVE_MESSAGE	NO NO NO
MEME	ME 0301 ME 0302	CONVERSE INQUIRE_MESSAGE INQUIRE_MESSAGE_LENGTH RETRIEVE_MESSAGE SEND_MESSAGE VALIDATE_LANGUAGE_CODE VALIDATE_LANGUAGE_SUFFIX	NO NO NO NO NO NO NO
MESR	ME 0201 ME 0202	SET_MESSAGE_OPTIONS	NO

MEBM gate, RETRIEVE_MESSAGE function

The RETRIEVE_MESSAGE function of the MEBM gate is used to retrieve the message text and build the message into a buffer.

Input parameters

MESSAGE_TABLE
is a table containing all the message definitions for the message domain.

[COMPONENT_ID]
is the component identifier for the message.

MESSAGE_NUMBER
is the numeric message identifier.

MESSAGE_BUFFER
is the buffer to receive the message text.

[INSERT1] through [INSERT10]
are user-supplied inserts, if these are required by the message definition.

[SYMPTOM_BUFFER]
is the buffer to receive a symptom string for the message.

[SUPPRESS_SRBUILD]
indicates whether or not a symptom record build is suppressed. It can have either of these values:

YES|NO

[MODULE_NAME]
is the name of the module in error, supplied as data for the symptom string.

[MODULE_PTF]
is the PTF level of the module in error, supplied as data for the symptom string.

[UPPERCASE]
determines whether or not messages should be converted to uppercase. It can have either of these values:

YES|NO

Output parameters

RESPONSE
is the domain's response to the call. It can have either of these values:

OK|EXCEPTION

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:

MESSAGE_CANNOT_BE_PRODUCED

MEBM gate, INQUIRE_MESSAGE_LENGTH function

The INQUIRE_MESSAGE_LENGTH function of the MEBM gate is used to find the length of the message in order to obtain the appropriate sized buffer to retrieve the message.

Input parameters

MESSAGE_TABLE
is a table containing all the message definitions for messages output by the message domain.

[COMPONENT_ID]
is the component identifier for the message.

MESSAGE_NUMBER
is the numeric message identifier.

[INSERT1] through [INSERT10]
are user-supplied inserts, if these are required by the message definition.

Output parameters

MESSAGE_LENGTH
is the length of the message being inquired on.

RESPONSE
is the domain's response to the call. It can have either of these values:

OK|EXCEPTION

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:

MESSAGE_CANNOT_BE_FOUND

MEBM gate, INQUIRE_MESSAGE_DEFINITION function

The INQUIRE_MESSAGE_DEFINITION function of the MEBM gate is used to return the action and severity codes of a message.

Input parameters

MESSAGE_TABLE
is a table containing all the message definitions for the message domain.

[COMPONENT_ID]
is the component identifier for the message.

MESSAGE_NUMBER
is the numeric message identifier.

Output parameters

SEVERITY_CODE
is the severity of the message.

ACTION_CODE
is the action code for the message.

RESPONSE
is the domain's response to the call. It can have either of these values:

OK|EXCEPTION

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:

MESSAGE_CANNOT_BE_FOUND

MEME gate, SEND_MESSAGE function

The SEND_MESSAGE function of the MEME gate is used to send a message to one or more destinations.

Input parameters

[COMPONENT_ID]
is the component identifier for the message.

MESSAGE_NUMBER
is the numeric message identifier.

[PRODUCT] is an optional product identifier.

[MSGTABLE] indicates that the feature message table is to be used.

[SYSTEM_DUMPCODE]
is the dump code to be used when the message domain requests a dump on behalf of its caller.

[TERMINATE_CICS]
specifies whether the caller requests CICS to be terminated.

[RESTART_CICS]
specifies whether the caller requests CICS to be automatically restarted by ARM.

[INSERT1] through [INSERT10]
are user-supplied inserts, if these are required by the message definition.

[TRANID] is the transaction identifier to be used to override the tranid obtained by the message domain.

[TERMID] is the terminal identifier to be used to override the termid obtained by the message domain.

[NETNAME] is the network name to be used to override the netname obtained by the message domain.

[NOREROUTE] specifies whether the caller requests that the message is to be rerouted to a transient data
destination.

[TDQUEUES] specifies the transient data destinations to which a message is to be sent.

[IGNORE_EXCEPTIONS]
specifies whether the caller requests that a failure sending a message to a transient data
destination is to be ignored.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, INVALID, or PURGED. Possible values are:

RESPONSE	Possible REASON values
DISASTER	INVALID_MODULE_PTR INVALID_TEMPLATE ABEND INSUFFICIENT_STORAGE
INVALID	INVALID_COMPONENT_TYPE INVALID_DBCS_FORMAT INVALID_DESTINATION INVALID_FUNCTION INVALID_INSERT INVALID_MEFO_RESPONSE MESSAGE_NOT_FOUND MESSAGE_SET_NOT_FOUND MISSING_INSERT OPT_INSERT_NOT_FOUND RETRY_MSG_LOCATE
PURGED	TDQ_PURGED

MEME gate, CONVERSE function

The CONVERSE function of the MEME gate is used to send a message and receive a reply.

Input parameters

[COMPONENT_ID]
is the component identifier for the message.

MESSAGE_NUMBER
is the numeric message identifier.

[PRODUCT]
is an optional product identifier.

[INSERT1] through [INSERT10]
are user-supplied inserts, if these are required by the message definition.

[TRANID]
is the transaction identifier to be used to override the tranid obtained by the message domain.

[TERMID]
is the terminal identifier to be used to override the termid obtained by the message domain.

[NETNAME]
is the network name to be used to override the netname obtained by the message domain.

[REPLY_BUFFER]
is the buffer into which the text reply is to be returned.

REPLY_FORMAT (VALUE|TEXT_OR_VALUE|TEXT)
indicates the format of the reply. It can be one of these formats:

VALUE|TEXT_OR_VALUE|TEXT

Output parameters

[REPLY_INDEX]
is the number of the template reply option that matches the user's reply text.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	INVALID_MODULE_PTR INVALID_TEMPLATE MAX_REPLIES_EXCEEDED ABEND INSUFFICIENT_STORAGE
EXCEPTION	REPLY_BUFFER_TOO_SMALL
INVALID	INVALID_COMPONENT_TYPE INVALID_DESTINATION INVALID_FUNCTION INVALID_INSERT INVALID_REPLY_BUFFER MESSAGE_NOT_FOUND MESSAGE_SET_NOT_FOUND MISSING_INSERT OPT_INSERT_NOT_FOUND REPLY_BUFFER_REQUIRED REPLY_INDEX_REQUIRED RETRY_MSG_LOCATE

MEME gate, RETRIEVE_MESSAGE function

The RETRIEVE_MESSAGE function of the MEME gate is used to retrieve a message text.

Input parameters

[COMPONENT_ID]
is the component identifier for the message.

MESSAGE_NUMBER
is the numeric message identifier.

MESSAGE_BUFFER
is the buffer to receive the message text.

[PRODUCT] is an optional product identifier.

[MSGTABLE] indicates that the feature message table is to be used.

[LANGUAGE] is an optional language code.

[INSERT1] through [INSERT10]
are user-supplied inserts, if these are required by the message definition.

[TRANID] is the transaction identifier to be used to override the tranid obtained by the message domain.

[TERMID] is the terminal identifier to be used to override the termid obtained by the message domain.

[NETNAME] is the network name to be used to override the netname obtained by the message domain.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INSUFFICIENT_STORAGE INVALID_MODULE_PTR INVALID_TEMPLATE
EXCEPTION	MSG_BUFFER_TOO_SMALL REPLY_BUFFER_TOO_SMALL
INVALID	INVALID_COMPONENT_TYPE INVALID_FUNCTION, INVALID_INSERT INVALID_MESSAGE_BUFFER MESSAGE_NOT_FOUND MESSAGE_SET_NOT_FOUND, MISSING_INSERT OPT_INSERT_NOT_FOUND RETRY_MSG_LOCATE

MEME gate, INQUIRE_MESSAGE_LENGTH function

The INQUIRE_MESSAGE_LENGTH function of the MEME gate is used to find the length of the message in order to obtain the appropriate size buffer to retrieve the message.

Input parameters

[COMPONENT_ID]
is the component identifier for the message.

MESSAGE_NUMBER
is the numeric message identifier.

[PRODUCT] is an optional product identifier.

[MSGTABLE] indicates that the feature message table is to be used.

[LANGUAGE] is an optional language code.

[INSERT1] through [INSERT10]
are user-supplied inserts, if these are required by the message definition.

Output parameters

MESSAGE_LENGTH
is the length of the message being inquired on.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INSUFFICIENT_STORAGE INVALID_MODULE_PTR INVALID_TEMPLATE
INVALID	INVALID_COMPONENT_TYPE INVALID_FUNCTION INVALID_INSERT MESSAGE_NOT_FOUND MESSAGE_SET_NOT_FOUND MISSING_INSERT OPT_INSERT_NOT_FOUND RETRY_MSG_LOCATE

MEME gate, VALIDATE_LANGUAGE_CODE function

The VALIDATE_LANGUAGE_CODE function of the MEME gate is used to determine whether a specific three-letter IBM standard national language code is valid. If it is valid, this function returns the equivalent one-character CICS language suffix. The IBM standard three-character codes, and their corresponding one-character CICS language suffices, are listed in Table 70 on page 907.

Input parameters

LANGUAGE_CODE
is the three-character national language code to be validated.

Output parameters

[LANGUAGE_SUFFIX]
is the one-character CICS language suffix that corresponds to the input LANGUAGE_CODE.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|EXCEPTION|INVALID

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	LANGUAGE_CODE_INVALID LANGUAGE_NOT_SUPPORTED
INVALID	INVALID_FUNCTION

Table 70. Languages and their codes

NATLANG code	NLS code	Language
A	ENG	Alternative English
Q	ARA	Arabic
1	BEL	Byelorussian
L	BGR	Bulgarian
B	PTB	Brazilian Portuguese
T DBCS	CHT	Traditional Chinese
C DBCS	CHS	Simplified Chinese
2	CSY	Czech
D	DAN	Danish
G	DEU	German
O	ELL	Greek
S	ESP	Spanish
W	FIN	Finnish
F	FRA	French
X	HEB	Hebrew
3	HRV	Croatian
4	HUN	Hungarian
J	ISL	Icelandic
I	ITA	Italian
H DBCS	KOR	Korean
M	MKD	Macedonian
9	NLD	Dutch
N	NOR	Norwegian
5	PLK	Polish
P	PTG	Portuguese
6	ROM	Romanian
R	RUS	Russian
Y	SHC	Serbo-Croatian (Cyrillic)
7	SHL	Serbo-Croatian (Latin)
V	SVE	Swedish
Z	THA	Thai
8	TRK	Turkish
U	UKR	Ukrainian

Notes:

1. **DBCS** denotes Double-Byte Character Set languages.
2. A for *alternative English*. Code letter A means “alternative English” to distinguish your edited English message tables from the default US English message tables supplied by CICS. The default US English tables are designated by the language code letter E.
3. The NATLANG code for the selected language is used as the suffix of your edited message data sets that you can create using the message editing utility. For more information about the message editing utility, see *CICS Operations and Utilities Guide*.

MEME gate, VALIDATE_LANGUAGE_SUFFIX function

The VALIDATE_LANGUAGE_SUFFIX function of the MEME gate is used to determine whether a specific one-character CICS language suffix is valid. If it is valid, this function returns the equivalent three-character IBM standard national language code. The IBM standard three-character codes, and their corresponding one-character CICS language suffices, are listed in Table 70.

Input parameters

LANGUAGE_SUFFIX
is the one-character CICS language code to be validated.

Output parameters

[LANGUAGE_CODE]
is the three-character CICS language suffix that corresponds to the input LANGUAGE_SUFFIX.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|EXCEPTION|INVALID

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
EXCEPTION	LANGUAGE_NOT_SUPPORTED LANGUAGE_SUFFIX_INVALID
INVALID	INVALID_FUNCTION

MEME gate, INQUIRE_MESSAGE function

The INQUIRE_MESSAGE function of the MEME gate is used to find the system default language as a one-character CICS language suffix and a three-character IBM standard national language code.

Input parameters

None.

Output parameters

DEFAULT_LANGUAGE_CODE
is the three-character code for the default language.

DEFAULT_LANGUAGE_SUFFIX
is the one-character suffix for the default language.

RESPONSE
is the domain's response to the call. It can have either of these values:

OK|DISASTER|INVALID

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND
INVALID	INVALID_FUNCTION

MESR gate, SET_MESSAGE_OPTIONS function

The SET_MESSAGE_OPTIONS function of the MESR gate is used to set the various message options specified by the system initialization parameters MSGCASE, MSGLVL, and NATLANG.

Input parameters

[LANGUAGES_USED]
is a list of the languages used in the system.

[MESSAGE_LEVEL]
can be 0 or 1. 0 means that information messages do not appear (are suppressed) at the console.

[MESSAGE_CASE]
is either MIXED for mixed-case messages, or UPPER for messages to be folded to uppercase.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. It has this value:

INVALID_FUNCTION

Message domain's generic gate

Table 71 summarizes the message domain's generic gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 71. Message domain's generic gate

Gate	Trace	Function	Format
DMDM	ME 0101 ME 0102	PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

You can find descriptions of these functions and their input and output parameters, in the section dealing with the corresponding generic formats, in "Domain manager domain's generic formats" on page 669.

In preinitialization processing, the message domain sets the following message options:
- The national languages to be supported during this CICS run
- The message level for initialization messages
- The message case.

For a cold start, the information comes from the system initialization parameters; for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

The message domain does no quiesce processing or termination processing.

Modules

Module	Function
DFHCMAC	Displays messages and codes online for the CMAC transaction
DFHMEBM	Is executed in an offline environment, and is provided for use by batch utility programs
DFHMEBU	Builds a message into a buffer, and also builds a symptom string when required
DFHMEDM	Performs the necessary domain manager functions; that is, preinitialize, initialize, quiesce, and terminate for the message domain
DFHMEDUF	ME domain offline dump formatting routine
DFHMEFO	Formats a long message into lines of specified length
DFHMEIN	Provides all the data required to build a message

Module	Function	
DFHMEME	Handles the following functions:	
	SEND_MESSAGE	sends a message to any individual or combination of MVS/MCS consoles, or CICS TD queues.
	CONVERSE	sends a message to any individual or combination of MVS/MCS consoles and receives a reply from one of them.
	RETRIEVE_MESSAGE	builds a message and places it in a buffer passed by the caller.
	INQUIRE_MESSAGE_LENGTH	returns the length of a terminal end user message.
	INQUIRE_MESSAGE	returns the requested data, held by the ME domain (for example, Default_Language).
	VALIDATE_LANGUAGE_CODE	checks whether a three-character language code is valid.
	VALIDATE_LANGUAGE_SUFFIX	checks whether a one-character language suffix is valid.
DFHMESR	Collects the system initialization parameter overrides for a particular CICS start	
DFHMETRI	ME domain offline trace interpretation routine	
DFHMEWS	Writes a symptom record containing a symptom string to SYS1.LOGREC by using the MVS SYMRBLD macro	
DFHMEWT	Provides support to execute the MVS WTOR SVC	

Exits

There is one global user exit point in the message domain: XMEOUT. See the *CICS Customization Guide* for further information.

Trace

The point IDs for the message domain are of the form ME xxxx; the corresponding trace levels are ME 1, ME 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 91. Monitoring domain (MN)

The monitoring domain is responsible for all monitoring functions within CICS. These functions enable the user to measure the amount of CPU, storage, temporary-storage requests, and so on used per task, and hence charge customers for computing services and help review the performance of a CICS system.

Monitoring domain's specific gates

Table 72 summarizes the monitoring domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 72. Monitoring domain's specific gates

Gate	Trace	Function	XPI
MNMN	MN 0201 MN 0202	EXCEPTION_DATA_PUT PERFORMANCE_DATA_PUT INQUIRE_MONITORING_DATA MONITOR INQUIRE_RESOURCE_DATA	NO NO YES* YES NO
MNSR	MN 0301 MN 0302	SET_MCT_SUFFIX SET_MONITORING INQ_MONITORING	NO NO NO
MNXM	MN 0A01 MN 0A02	TRANSACTION_INITIALIZATION TRANSACTION_TERMINATION	NO NO NO

* In a modified form, without a transaction number or current data buffer.

MNMN gate, EXCEPTION_DATA_PUT function

The EXCEPTION_DATA_PUT function of the MNMN gate is used to produce an exception record at the completion of an EXCEPTION condition.

Input parameters

EXCEPTION_START
is the start time of the exception in stored clock (STCK) format.

EXCEPTION_STOP
is the stop time of the exception in STCK format.

RESOURCE_TYPE
is the type of resource for which the exception data is to be recorded.

RESOURCE_ID
is the identifier of the resource for which the exception data is to be recorded.

EXCEPTION_TYPE
is the type of exception to be recorded.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INVALID_MONITORING_TOKEN LOOP

MNMN gate, PERFORMANCE_DATA_PUT function

The PERFORMANCE_DATA_PUT function of the MNMN gate is used to produce a performance record and reset task monitoring information for a conversational task or a syncpoint.

Input parameters

RECORD_TYPE is the reason for the record to be output.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INVALID_MONITORING_TOKEN LOOP

MNMN gate, INQUIRE_MONITORING_DATA function

The INQUIRE_MONITORING_DATA function of the MNMN gate is used to access a transaction's monitoring information.

Input parameters

[TRANSACTION_NUMBER]
is the transaction number for which monitoring data is required.

DATA_BUFFER specifies the address and length of a buffer for the monitoring data.

[CURRENT_DATA_BUFFER]
specifies the address and length of a buffer for the current monitoring data.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	MONITOR_DATA_UNAVAILABLE LENGTH_ERROR

MNMN gate, MONITOR function

The MONITOR function of the MNMN gate is called to process a user event-monitoring point (EMP).

Input parameters

POINT is a value in the range 0 through 255 corresponding to a monitoring point identifier defined in the monitoring control table (MCT).

[ENTRYNAME] is an ID qualifier, 1 through 8 bytes, corresponding to an entry name specified in the MCT.

[DATA1] supplies 4 bytes of data to be used in the operations performed by this user's EMP.

[DATA2] supplies 4 bytes of data to be used in the operations performed by this user's EMP.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INVALID_MONITORING_TOKEN LOOP
EXCEPTION	POINT_NOT_DEFINED DATA1_NOT_SPECIFIED DATA2_NOT_SPECIFIED INVALID_DATA1_VALUE INVALID_DATA2_VALUE

MNMN gate, INQUIRE_RESOURCE_DATA function

The INQUIRE_RESOURCE_DATA function of the MNMN gate is used to access a transaction's resource data when transaction resource monitoring is active.

Input parameters

[TRANSACTION_NUMBER]
is the transaction number for which transactoin resource data is required.

RESOURCE_DATA_BUFFER
specifies the address and length of a buffer for the transaction resource data.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	RESOURCE_DATA_UNAVAILABLE LENGTH_ERROR

MNMN gate, ACCUMULATE_RMI_TIME function

The ACCUMULATE_RMI_TIME function of the MNMN gate is used to accumulate all of the appropriate performance class DFHRMI timing fields.

Input parameters

[TRUE_NAME]
is the name of the CICS resource manager being used by your transaction.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND, LOOP
EXCEPTION	INVALID_MONITORING_TOKEN

MNSR gate, SET_MCT_SUFFIX function

The SET_MCT_SUFFIX function of the MNSR gate is used to identify to the monitoring domain the suffix of the monitoring control table (MCT).

Input parameters

SUFFIX
is the 2-character MCT suffix.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	MCT_NOT_FOUND

MNSR gate, SET_MONITORING function

The SET_MONITORING function of the MNSR gate is used to set the monitoring classes on or off and to change the monitoring options.

Input parameters

[CONVERSE]
indicates if a transaction performance class record is to be produced for conversational tasks for each pair of terminal control I/O requests. It can have either of these values:

YES|NO

[EXCEPTION_STATUS]
indicates the exception class monitoring setting. It can have either of these values:

ON|OFF

[FREQUENCY]
is the interval for which monitoring automatically produces a transaction performance class record for any long-running transaction. Frequency times are 0, or in the range 000100 through 240000. The default frequency value is 0, which means that frequency monitoring is inactive.

[MONITORING_STATUS]
indicates the monitoring status setting. It can have either of these values:

ON|OFF

[PERFORMANCE_STATUS]
indicates the performance class monitoring setting. It can have either of these values:

ON|OFF

[RESOURCE_STATUS]
indicates the transaction resource class monitoring setting. It can have one of the following values:

ON|OFF

[SUBSYSTEM_ID]
specifies the 4-character subsystem-id to be used in the MVS workload activity records. The default is the first four character of the generic applid.

[SYNCPOINT]
indicates if a transaction performance class record is to be produced when a transaction takes an explicit or implicit syncpoint (unit-of-work). It can have either of these values:

YES|NO

[TIME]
indicates whether the monitoring timestamp fields returned on the INQUIRE_MONITORING_DATA function are to be in GMT or Local time. It can have either of these values:

GMT|LOCAL

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	INVALID_FREQUENCY

MNSR gate, INQ_MONITORING function

The INQ_MONITORING function of the MNSR gate is used to enquire on the monitoring classes and the monitoring options.

Input parameters

None.

Output parameters

CONVERSE
indicates if a transaction performance class record is to be produced for conversational tasks for each pair of terminal control I/O requests. It can have either of these values:

YES|NO

EXCEPTION_STATUS
indicates whether exception class monitoring is active. It can have either of these values:

ON|OFF

FREQUENCY
is the interval for which monitoring automatically produces a transaction performance class record for any long-running transaction. Frequency times are 0, or in the range 000100 through 240000. The default frequency value is 0, which means that frequency monitoring is inactive.

MONITORING_STATUS
indicates whether monitoring is active. It can have either of these values:

ON|OFF

PERFORMANCE_STATUS
indicates whether performance class monitoring is active. It can have either of these values:

ON|OFF

RESOURCE_STATUS
indicates whether transaction resource class monitoring is active. It can have one of the the following values:

ON|OFF

SUBSYSTEM_ID
specifies the 4-character subsystem-id to be used in the MVS workload activity records. The default is the first four character of the generic applid. implicit syncpoint (unit-of-work).

SYNCPOINT
indicates if a transaction performance class record is to be produced when a transaction takes an explicit or implicit syncpoint (unit-of-work). It can have either of these values:

YES|NO

TIME
indicates whether the monitoring timestamp fields returned on the INQUIRE_MONITORING_DATA function are to be in GMT or Local time. It can have either of these values:

GMT|LOCAL

APPLICATION_NAMING
indicates whether application naming support is enabled in the CICS region. It can have either of these values:

YES|NO

RMI_STATUS
indicates whether additional monitoring performance class data is required for the resource managers used by your transaction. It can have either of these values:

YES|NO

FILE_LIMIT
specifies the maximum number of files for which you want CICS to perform transaction resource monitoring. It can have a value in the range 1 through 64.

TSQUEUE_LIMIT
specifies the maximum number of temporary storage queues for which you want CICS to perform transaction resource monitoring. It can have a value in the range 1 through 32.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	SUBSYSTEM_ID_NOT_AVAILABLE

MNXM gate, TRANSACTION_INITIALIZATION function

The TRANSACTION_INITIALIZATION function of the MNXM gate is used to inform the monitoring domain of a transaction attach request so that the monitoring domain can allocate task monitoring storage.

Input parameters

TASK_ATTACH_TIME
is the time when this task was attached.

TCLASS_DELAY_TIME
is the time this task was delayed due to the transaction class (if any) limit for this transaction being reached.

MXT_DELAY_TIME
is the time this task was delayed due to the maximum user task limit (MXT) being reached.

INITIAL_DISPATCH_TIME
is the time when this task was first dispatched after attach.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INVALID_MONITORING_TOKEN LOOP

MNXM gate, TRANSACTION_TERMINATION function

The TRANSACTION_TERMINATION function of the MNXM gate is used to inform the monitoring domain of a transaction detach request, so that the monitoring domain can report on task monitoring information and then release the storage.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INVALID_MONITORING_TOKEN LOOP

Monitoring domain's generic gates

Table 73 summarizes the monitoring domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 73. Monitoring domain's generic gates

Gate	Trace	Function	Format
APUE	MN 0601 MN 0602	SET_EXIT_STATUS	APUE
DMDM	MN 0101 MN 0102	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	MN 0401 MN 0402	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
TISR	MN 0801 MN 0802	NOTIFY	TISR
XMNT	MN 0901 MN 0902	MXT_CHANGE_NOTIFY	XMNT

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format APUE—"Application domain's generic formats" on page 593

Format DMDM—"Domain manager domain's generic formats" on page 669

Format STST—"Statistics domain's generic format" on page 1198

Format TISR—"Timer domain's generic format" on page 1203

Format XMNT—"Transaction manager domain's generic format" on page 1308.

In initialization processing, the monitoring domain sets the initial monitoring options:
- Monitoring control table suffix
- Initial monitoring status
- Initial exception class monitoring status
- Initial performance class monitoring status
- Initial transaction resource class monitoring status
- Initial converse option
- Initial syncpoint option
- Initial time option
- Initial frequency option
- Initial subsystem id.

For a cold start, the information comes from the system initialization parameters; for any other type of start, the information comes from the global catalog, but is then modified by any relevant system initialization parameters.

In addition:
- If necessary, the monitoring control table (MCT) is loaded and initialized.

- If performance class monitoring is active, CPU timing is started.
- The monitoring domain user exit gate is enabled.
- Messages are sent to the console to indicate whether monitoring is active, and what MCT suffix (if any) is being used.

In quiesce processing, the monitoring domain waits for all transactions that it is monitoring to terminate. Then the final data in the performance class buffer and the transaction resource class buffer, if any, is written to SMF.

The monitoring domain does no termination processing.

Modules

Module	Function
DFHMNDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHMNDUF	Formats the MN domain control blocks in a CICS system dump
DFHMNMN	Handles the following requests: EXCEPTION_DATA_PUT PERFORMANCE_DATA_PUT INQUIRE_MONITORING_DATA MONITOR INQUIRE_RESOURCE_DATA ACCUMULATE_RMI_TIME
DFHMNNT	Handles the following request: MXT_CHANGE_NOTIFY
DFHMNSR	Handles the following requests: SET_MCT_SUFFIX SET_MONITORING INQ_MONITORING
DFHMNST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHMNSU	Handles monitoring domain subroutine requests of format MNSU: • UPDATE_CATALOGUE • MONITORING_DATASET_PUT • WLM_CONNECT • WLM_DISCONNECT • WLM_REPORT • WLM_NOTIFY • PB_ALLOCATE • PB_DELETE
DFHMNSVC	Provides SMFEWTM, WLM_CONNECT, WLM_DISCONNECT, WLM_REPORT, WLM_NOTIFY, WLM_PB_CREATE, and WLM_PB_DELETE authorized services with GTF tracing (GTRACE)
DFHMNTI	Handles the following request: NOTIFY
DFHMNTRI	Provides a trace interpretation routine for CICS dumps and traces
DFHMNUE	Provides a SET_EXIT_STATUS (services user exit) routine to enable or disable an exit
DFHMNXM	Handles the following requests: TRANSACTION_INITIALIZATION TRANSACTION_TERMINATION

Exits

There is one global user exit point in the monitoring domain: XMNOUT. See the *CICS Customization Guide* for further information.

Trace

The point IDs for the monitoring domain are of the form MN xxxx; the corresponding trace levels are MN 1, MN 3, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 92. Enqueue Domain (NQ)

The NQ domain provides UOW based locking services. This is provided to the local clients FC, TD and TS. It also services the EXEC CICS ENQ/DEQ requests.

The most common functions provided by the NQ domain are:

CREATE_ENQUEUE_POOL
This function creates a separate enqueue pool for the caller. A token is returned which the caller specifies on all requests associated with that pool.

DEACTIVATE
This function converts an active enqueue into retained state. The caller must already own the enqueue.

REACQUIRE_ENQUEUE
NQ domain doesn't recover enqueues over a CICS restart. Instead resource owners use this function to reacquire enqueues that were held by inflight and indoubt UOWs.

ENQUEUE
This functions obtains an enqueue from the specified enqueue pool in active state.

DEQUEUE
This functions releases an active enqueue owned by the current UOW from the specified enqueue pool.

INQUIRE_NQRNAME
This function calls INQ_NQRNAME to see if an enqueue name entry exists in NQRNAME_LIST. If the name is either an exact or generic match, INQUIRE_NQRNAME returns the 4-character SCOPE name, enqmodel STATE and ann OK RESPONSE. Otherwise it returns an EXCEPTION REASON(NQRNAME_NOT_FOUND).

ADD_REPLACE_ENQMODEL
This function adds an enqmodel definition to both the NQRN directory (keyed by enqmodel name, and to the NQRNAME_LIST (keyed by the variable length NQRNAME). If the enqmodel already exists the entry is replaced.

DISCARD_ENQMODEL
Remove an enqmodel definition from both the NQRN directory and from the NQRNAME_LIST. If the enqmodel is not installed, exception 'ENQMODEL_NOT_FOUND' is returned.

INQUIRE_ENQMODEL
Uses directory DDLO_LOCATE to retrieve information about a specified enqmodel definition in the NQRN directory.

If found, it returns the 1 to 255 character NQRNAME, the 4-character SCOPE name, the enqmodel STATE and ann OK RESPONSE. Otherwise it returns an EXCEPTION REASON(ENQMODEL_NOT_FOUND).

SET_ENQMODEL
This function uses directory DDLO_LOCATE to see if an enqmodel entry exists in the NQRN directory. If found, it enables or disables the entry. Otherwise it returns an EXCEPTION REASON(ENQMODEL_NOT_FOUND).

Enqueue domain's specific gates

Table 74 summarizes the NQ domain's specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate and the functions provided by the gate. The DFHNQEDX XPI macro provides ENQUEUE and DEQUEUE functions for the NQ domain.

Table 74. NQ domain’s specific gates

Gate	Trace	Function
NQNQ	NQ 0201 NQ 0202	CREATE_ENQUEUE_POOL DEACTIVATE REACQUIRE_ENQUEUE SET_NQRNAME_LIST DEQUEUE_TASK
NQED	NQ 0301 NQ 0302	ENQUEUE DEQUEUE

Table 74. NQ domain's specific gates (continued)

Gate	Trace	Function
NQIB	NQ 0401 NQ 0402	INQUIRE_ENQUEUE START_BROWSE_ENQUEUE GET_NEXT_ENQUEUE END_BROWSE_ENQUEUE
NQRN	NQ 0601 NQ 0602	INQUIRE_NQRNAME ADD_REPLACE_ENQMODEL DISCARD_ENQMODEL REMOVE_ENQMODEL INQUIRE_ENQMODEL START_BROWSE_ENQMODEL GET_NEXT_ENQMODEL END_BROWSE_ENQMODEL SET_ENQMODEL COMMIT_ENQMODEL RESTORE_DIRECTORY
NQIE	NQ FF50 NQ FF51	INTERPRET_ENQUEUE

NQNQ gate, CREATE_ENQUEUE_POOL function

This function creates a separate enqueue pool for the caller. A token is returned which the caller specifies on all requests associated with that pool.

Input parameters:

POOL_NAME
The eight character name of the new enqueue pool.

EXPECTED_NAME_LENGTH
The expected length for enqueue names in the pool.

For pools with fixed length enqueue names this should be the length of the names that are going to be enqueued upon.

For pools that are to contain variable length enqueue names this should be a length that would satisfy 'most' of the requests to be made in the pool.

Note that is no maximum length for enqueue names. However, requests will only be handled inline if the length of the enqueue name is less than or equal to the EXPECTED_NAME_LENGTH. The inline macro only copes with names of less than or equal to 256 characters. For this reason an error will be diagnosed if a value of greater than 256 is specified for this parameter.

SHUNT_ACTION
Indicates the **default** action that is to be performed to UOW lifetime enqueues in this pool if their owning UOW is shunted. Note that most enqueue pools will require the same action to be performed for all enqueues in that pool. However, the ENQUEUE function allows this default to be overridden for particular enqueue requests.

The possible values are as follows:

RELEASE
The enqueue(s) will be released if the owning UOW is shunted.

RETAIN
The enqueue(s) will be retained if the owning UOW is shunted.

IGNORE
The shunt will be ignored. The enqueue(s) will remain in the same state as currently held in.

Transaction lifetime enqueues are automatically released when a shunt occurs.

ERROR_LEVEL
Indicates the severity of the error response that is to be returned for the following errors made while using this pool:

- DEQUEUE
  - Enqueue_not_owned
  - Enqueue_locked
- REACQUIRE_ENQUEUE

  - Enqueue_locked
  - Enqueue_active
- DEACTIVATE
  - Enqueue_not_owned
  - Enqueue_not_active

The possible values for ERROR_LEVEL are as follows:

EXCEPTION_RESPONSE
The above errors are to be returned with an exception response.

INVALID_RESPONSE
The above errors are to be returned with an invalid response. (i.e. FFDC is to be performed).

Note: It is expected that only the EXEC and the KC enqueue pools will specify EXCEPTION_RESPONSE since the DFHKC service previously used by them allowed these sorts of error to go by undetected.

EXEC_INTERPRETER
Indicates how enqueues belonging to the enqueue pool are to be interpreted by the EXEC CICS INQUIRE UOWENQ command.

The possible values are as follows:

NONE
No interpreter has been supplied so enqueues belonging to this pool will be ignored by the INQUIRE UOWENQ command.

DEFAULT
Enqueues are to be returned by the INQUIRE UOWENQ command. The default NQ domain interpreter will be called to perform the interpretation. This will map the outputs of the INQUIRE UOWENQ command as follows:

TYPE
Will be the CVDA corresponding to the ENQUEUE_TYPE parameter supplied on this call.

RESOURCE
Will be ENQUEUE_NAME1 as supplied on the NQED_ENQUEUE function.

QUALIFIER
Will be ENQUEUE_NAME2 if supplied on the NQED_ENQUEUE function. If not then no QUALIFIER data will be returned.

OWN
Enqueues are to be returned by the INQUIRE UOWENQ command. A routine provided by the pool owner will perform the interpretation. In this case the entry point of the routine must be passed in the INTERPRETER_ADDR parameter.

Note: The routine will be called by a kernel subroutine call, not by a domain call. Consequently it will execute in the domain of the caller (i.e. AP domain).

OWN_INTERPRETER_ADDRESS
Entry point of interpreter routine for this pool. Should only be supplied for pools which specify a value of OWN for the EXEC_INTERPRETER parameter.

ENQUEUE_TYPE
The enqueue type that is to be returned by the default interpreter. Should only be supplied for pools which specify a value of DEFAULT for the EXEC_INTERPRETER parameter.

The possible values are as follows and these map onto the CVDA values for the TYPE field as detailed under the EXEC CICS INQUIRE UOWENQ command.

- DATASET
- EXECENQ
- EXECENQADDR
- EXECENQPLEX
- FILE
- TDQUEUE

- TSQUEUE
- DISPATCHER

Output parameters:

POOL_TOKEN Token returned which identifies the newly created enqueue pool.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INTERPRETER_ADDR_EXPECTED ENQUEUE_TYPE_EXPECTED DUPLICATE_POOL_NAME INVALID_NAME_LENGTH

NQNQ gate, DEACTIVATE function

This function converts an active enqueue into retained state. The caller must already own the enqueue.

Input parameters:

POOL_TOKEN Token representing enqueue pool from which the enqueue is to be deactivated.

ENQUEUE_TOKEN
Token representing the enqueue that is to be deactivated.

Slightly better performance is achieved for callers that use the token method for this function.

ENQUEUE_NAME1
A block (addr,len) identifying the name of the enqueue to be deactivated.

Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name of the enqueue to be deactivated.

ENQUEUE_NAME2
A block (addr,len) identifying the second half of the enqueue name.

Output parameters:

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	ENQUEUE_NOT_OWNED ENQUEUE_NOT_ACTIVE
INVALID	ENQUEUE_NOT_OWNED ENQUEUE_NOT_ACTIVE TRANSACTION_ENQUEUE INVALID_POOL_TOKEN

NQNQ gate, REACQUIRE_ENQUEUE function

NQ domain doesn't recover enqueues over a CICS restart. Instead resource owners use this function to reacquire enqueues that were held by inflight and indoubt UOWs.

The enqueue can be reacquired in either active or retained state. The calling UOW must currently be shunted.

No MAX_LIFETIME input is provided since such enqueues are only ever associated with a single UOW.

The same rules as documented for the mainline ENQUEUE function apply to the shunt action that will be associated with the reacquired enqueue.

Input parameters:

POOL_TOKEN
Token representing enqueue pool from which the enqueue is to be allocated from.

ENQUEUE_NAME1
A block (addr,len) identifying the name of the enqueue.

Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name being enqueued on.

ENQUEUE_NAME2
A block (addr,len) identifying the second half of the enqueue name.

STATE
The state that the enqueue is to be reacquired in.

The possible states are as follows:

ACTIVE
The enqueue is to be reacquired in active state.

RETAINED
The enqueue is to be reacquired in retained state.

SHUNT_ACTION
Indicates the action that is to be performed if the UOW reacquiring the enqueue is shunted again. This parameter acts as an override, if not supplied then the default shunt action specified when the pool was created is assumed for this request.

The possible overrides are as follows:

RELEASE
The enqueue will be released if the UOW is shunted again.

RETAIN
The enqueue will be retained if the UOW is shunted again.

IGNORE
The shunt will be ignored. The enqueue will remain in the same state as it is currently held in.

Output parameters:

ENQUEUE_TOKEN
Token returned to represent the enqueue that has been successfully reacquired.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	ENQUEUE_LOCKED ENQUEUE_ACTIVE
INVALID	ENQUEUE_LOCKED ENQUEUE_ACTIVE CALLER_NOT_SHUNTED INVALID_POOL_TOKEN

NQNQ gate, SET_NQRNAME_LIST function

This function is called from three places in dfhnqrn:

discard_enqmodel
IF nqrmodel delete is set THEN the specified nqrmodel is removed from nqrname_list.

Add_replace_enqmodel
IF nqrmodel add is set THEN the specified nqrmodel is added to nqrname_list.

set_nqrmodel
IF neither delete or add is set THEN the specified nqrmodel is set disabled.

Input parameters:

MODEL_TOKEN
The address of the nqrmodel to be set or added to nqrname_list.

POOL_TOKEN
The pool to be searched for matching enqueues

POOL_TWO
An optional second pool to be searched for matching enqueues

Output parameters:

FREE_TOKEN
Address of Model being removed.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION or PURGED. Possible values are:

NQRMODEL_NOT_FOUND
The nqrmodel could not be found in nqrname_list

FREE_NQRMODEL
A nqrmodel has been removed and must be freemained. Its address is in free_token.

NQED gate, ENQUEUE function

This functions obtains an enqueue from the specified enqueue pool in active state.

Input parameters:

POOL_TOKEN
Token representing enqueue pool from which the enqueue is to be allocated.

ENQUEUE_NAME1
A block (addr,len) identifying the name being enqueued on.

Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name being enqueued on.

ENQUEUE_NAME2
A block (addr,len) identifying the second half of the enqueue name.

MAX_LIFETIME
Indicates the maximum duration that the enqueue is to be held for. The possible values are as follows:

UOW
The enqueue will be released if it is held when the current UOW commits. This is the default value when not supplied on the call.

TRANSACTION
The enqueue will be released if it is held when the last UOW in the current transaction commits.

DISPATCHER_TASK
The enqueue will be released if it is held when a DEQUEUE_ALL request is issued by the owning dispatcher task. This is the only value permitted when POOL_TOKEN is not supplied on the call.

WAIT
Indicates whether the caller wishes to wait if the requested enqueue is currently held in the pool by a different UOW. The possible values are as follows:

YES
The caller will be suspended if the enqueue is busy. This is the default value when not supplied on the call.

NO
The ENQUEUE_BUSY exception is returned to the caller if the enqueue is busy.

Note that callers specifying WAIT(NO) should still expect to suspend for the NQ domain lock.

SHUNT_ACTION
Indicates the action that is to be performed if this UOW is shunted whilst it owns the

enqueue. This parameter acts as an override, if not supplied then the default shunt action specified when the pool was created is assumed for this enqueue request.

The shunt action is only applicable to UOW lifetime enqueues. An error is diagnosed if this parameter is supplied on a request for a transaction lifetime enqueue.

The possible overrides are as follows:

RELEASE
The enqueue will be released if the UOW is shunted.

RETAIN
The enqueue will be retained if the UOW is shunted.

IGNORE
The shunt will be ignored. The enqueue will remain in the same state as it is currently held in.

Output parameters:

ENQUEUE_TOKEN
Token returned to represent the enqueue that has been successfully returned.

The token can then be used on the corresponding DEQUEUE request.

DUPLICATE_REQUEST
When an OK is returned this indicates whether the caller already owned the enqueue or not:

YES
The caller already owned the enqueue.

NO The caller didn't already own the enqueue.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, PURGED or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	ENQUEUE_BUSY ENQUEUE_LOCKED ENQUEUE_DISABLED LIMIT_EXCEEDED SYSENQ_FAILURE
PURGED	TASK_CANCELLED TIMED_OUT
INVALID	SHUNT_ACTION_NOT_EXPECTED INVALID_POOL_TOKEN

NQED gate, DEQUEUE function

This functions releases an active enqueue owned by the current UOW from the specified enqueue pool.

Input parameters:

POOL_TOKEN Token representing enqueue pool from which the enqueue is to be released.

ENQUEUE_TOKEN
Token representing the enqueue that is to be released.

Slightly better performance is achieved for callers that use the token method for releasing their enqueues.

ENQUEUE_NAME1
A block (addr,len) identifying the name of the enqueue being released.

Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name of the enqueue being released.

ENQUEUE_NAME2
A block (addr,len) identifying the second half of the enqueue name.

MAX_LIFETIME
Indicates the maximum duration of the enqueue being released. The possible values are as follows:

UOW
The enqueue was acquired with a duration of the current UOW. This is the default value when not supplied on the call.

TRANSACTION
The enqueue was acquired with a duration of the last UOW of the current transaction.

Output parameters:

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	ENQUEUE_NOT_OWNED ENQUEUE_LOCKED
INVALID	ENQUEUE_NOT_OWNED ENQUEUE_LOCKED INVALID_POOL_TOKEN

NQIB gate, INQUIRE_ENQUEUE function

This functions returns information about a particular enqueue. Note that the pool containing the enqueue must be passed since it is a logical extension to the enqueue name.

For inquiries by token it is the caller's responsibility to ensure that the enqueue which the token represents is still held.

Input parameters:

POOL_TOKEN
The token identifying the pool from which the enqueue being inquired about belongs.

ENQUEUE_TOKEN
Token representing the enqueue that is being inquired upon.

ENQUEUE_NAME1
A block (addr,len) identifying the name of the enqueue be inquired upon.

Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name of the enqueue being inquired upon.

ENQUEUE_NAME2
A block (addr,len) identifying the second half of the enqueue name.

Output parameters:

ENQUEUE_NAME_OUT
A buffer into which the enqueue name is returned. The caller specifies the address and maximum length of the data area into which the enqueue name will be returned. If the enqueue name is too big for the buffer then the data is truncated and an OK response is returned. The actual length of the name is returned in enqueue_name_out_n.

Typically this parameter will only be of interest to callers inquiring by enqueue token.

POOL_NAME
The name of the pool containing the enqueue.

STATE
The state that the enqueue is held in.

ACTIVE
The enqueue is held in active state.

RETAINED
The enqueue is held in retained state.

LOCAL_UOWID The local UOWID of the UOW which owns the enqueue

UOW_LIFETIME The number of times the enqueue is held with UOW lifetime.

TRANSACTION_LIFETIME
The number of times the enqueue is held with TRANSACTION lifetime.

NUM_WAITERS The number of transactions waiting for this enqueue.

NUM_LOCKED_FAILURES
Returns the number of failed requests for this enqueue whilst it is held in retained state.

SHUNT_ACTION The action that would be performed to this enqueue should its owning UOW be shunted.

The possible values are as follows:

RELEASE
The enqueue will be released.

RETAIN
The enqueue will be retained.

IGNORE
The shunt will be ignored and the enqueue will remain in the same state.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	ENQUEUE_NOT_FOUND
INVALID	INVALID_POOL_TOKEN

NQIB gate, START_BROWSE_ENQUEUE function

This function initiates a browse of all enqueues currently in the system or currently associated with a given UOW.

The browse returns both enqueue owners and enqueue waiters. The RELATION output parameter on GET_NEXT_ENQUEUE indicates whether the data being returned is associated with the enqueue owner or a UOW waiting for that enqueue.

When a system wide browse is initiated the first enqueue in the system is returned with RELATION(OWNER). If the enqueue has any waiters then the same enqueue will be returned again for each of the waiters but this time with RELATION(WAITER). The data returned will be that associated with that particular waiter. After the last waiter has been returned the next owned enqueue will be returned.

If the browse is restricted to only a particular UOW then only the enqueues that UOW owns will be returned. If the UOW is waiting for an enqueue this will also be returned.

The order in which the enqueues are returned is undefined, however enqueue waiters are always returned consecutively after their enqueue owner

As with other types of CICS browses the state isn't locked for the duration of the browse. Thus for example, there is no guarantee that the owner returned on a previous GET_NEXT_ENQUEUE is still the owner by the time each of its waiters are returned.

Input parameters:

LOCAL_UOWID
Identifies the unit of work if the browse is to be restricted to only those enqueues owned and being waited for by a particular UOW.

If omitted then browse will return all enqueue owners and waiters in the system.

STABLE_ENQUEUES
Specifies that the caller will complete the browse without issuing any further ENQ or DEQ requests. Applies only if LOCAL_UOWID is also specified and names the caller's own UOWID.

Output parameters:

BROWSE_TOKEN
Token to be used by the caller on subsequent operations associated with this browse.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	NO_UOW_ENVIRONMENT

NQIB gate, GET_NEXT_ENQUEUE function

This functions returns information about the next enqueue owner or waiter in a browse.

Input parameters:

BROWSE_TOKEN
The token for the current browse.

Output parameters:

ENQUEUE_NAME_OUT
A buffer into which the enqueue name is returned. The caller specifies the address and maximum length of the data area into which the enqueue name will be returned. If the enqueue name is too big for the buffer then the data is truncated and an OK response is returned. The actual length of the name is returned in enqueue_name_out_n.

RELATION
Indicates whether the data being returned is associated with owner or a UOW waiting for the enqueue.

OWNER
The data is associated with the owner of the returned enqueue.

WAITER
The data is associated with a waiter of the returned enqueue.

POOL_NAME
The name of the pool containing the enqueue.

STATE
The state that the enqueue is held in.

ACTIVE
The enqueue is held in active state.

RETAINED
The enqueue is held in retained state.

LOCAL_UOWID
The local UOWID of the UOW which owns or is waiting for the enqueue.

UOW_LIFETIME
For an enqueue returned with RELATION(OWNER) the number of times it is held with UOW lifetime.

For an enqueue returned with RELATION(WAITER) a count of one indicates that the enqueue was requested with UOW lifetime.

TRANSACTION_LIFETIME
For an enqueue returned with RELATION(OWNER) the number of times it is held with TRANSACTION lifetime.

For an enqueue returned with RELATION(WAITER) a count of one indicates that the enqueue was requested with TRANSACTION lifetime.

NUM_WAITERS The number of transactions waiting for this enqueue.

NUM_LOCKED_FAILURES
Returns the number of failed requests for this enqueue whilst it is held in retained state.

SHUNT_ACTION The action that would be performed to this enqueue should its owning UOW be shunted.

The possible values are as follows:

RELEASE
The enqueue will be released.

RETAIN
The enqueue will be retained.

IGNORE
The shunt will be ignored and the enqueue will remain in the same state.

INTERPRETER_ADDRESS
The address of a routine which should be called with the INTERPRET_ENQUEUE function in order to interpret the enqueue for the EXEC CICS INQUIRE UOWENQ command.

If a zero address is returned then the enqueue isn't to be returned by the INQUIRE UOWENQ command.

POOL_TOKEN Token which identifies the pool which the enqueue owner or waiter belongs.

ENQUEUE_NAME2_LENGTH
The length of the second part of the enqueue name if the enqueue was originally specified in two parts (i.e. ENQUEUE_NAME1 and ENQUEUE_NAME2).

If the ENQUEUE_NAME2 parameter wasn't originally specified for this enqueue then zero will be returned.

ENQUEUE_TOKEN
Token returned only when the enqueue is owned by the caller. Parameter is set to zero for all other enqueues returned on the browse.

RESOURCE_FILTER
The resource filter as specified in the RESOURCE option on the ENQUIRE UOWENQ command.

RESOURCE_FILTER_LEN
The length of the RESOURCE_FILTER parameter.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	BROWSE_END
INVALID	INVALID_BROWSE_TOKEN

NQIB gate, END_BROWSE_ENQUEUE function

This functions terminates a browse of the enqueues.

Input parameters:

BROWSE_TOKEN The token for the browse that is to be terminated.

Output parameters:

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_BROWSE_TOKEN

NQRN gate, ENQUEUE function

This function calls INQ_NQRNAME to see if an enqueue name entry exists in NQRNAME_LIST.

If the name is either an exact or generic match, INQUIRE_NQRNAME returns the 4-character SCOPE name, enqmodel STATE and ann OK RESPONSE. Otherwise it returns an EXCEPTION REASON(NQRNAME_NOT_FOUND).

Input parameters:

NQRNAME A buffer giving a 1 to 255 char name and length of the resource to be located.

MSG0105 YES|NO, indicating whether message DFHNQ0105 is to be issued if the matching enqmodel is disabled or in the waiting state.

Output parameters:

SCOPE The 4-character scope identifier for the resource. Four blanks indicates that the enqueue has local scope.

STATE

ENABLED
Matching ENQ/DEQ requests should be processed.

DISABLED
Matching ENQ/DEQ requests should be rejected, and the issuing task abended abcode ENQ_DISABLED.

WAITING
Matching ENQ/DEQ requests should be rejected, and the issuing task abended abcode ENQ_DISABLED. There are INSTALL, CREATE, or DISCARD requests waiting to be processed.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, PURGED or INVALID. Possible values are:

NQRNAME_NOT_FOUND
The name does not exist in the table.

ACQUIRE_LOCK_FAILED
Attempt to acquire a shared NQRNAME lock failed.

RELEASE_LOCK_FAILED
Attempt to release a shared NQRNAME lock failed.

NQRN gate, ADD_REPLACE_ENQMODEL function

This function adds an enqmodel definition to both the NQRN directory (keyed by enqmodel name, and to the NQRNAME_LIST (keyed by the variable length NQRNAME).

If the enqmodel already exists the entry is replaced. The replace is a discard then add operation.

If an attempt is made to create a deep enqmodel nesting, or if another enqmodel with the same nqrname is already installed, then msg NQ0106 is issued and a 'DUPLICATE_NQRNAME' exception is returned.

Input parameters:

CALLER COLDINST, RDOINST or RESTART indicating A cold start, An online install or The input is in the MODEL_TOKEN respectively.

CATALOG YES or NO indicating whether the record should be cataloged.

ENQMODEL The 8-character identifier of the resource to be added.
MODEL_TOKEN The address of the record obtained from the catalogue to be restored.
SCOPE The 4-character scope identifier for the resource. If ommitted or specified as blanks, matching ENQs will have LOCAL scope.
STATE ENABLED|DISABLED is the state in which to install the enqmodel. If ommitted, ENABLED is assumed.
NQRNAME A buffer giving the 1 to 255 character name and length of the ENQ name or stem* to be added.

Output parameters:

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|PURGED
[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, PURGED or INVALID. Possible values are:
INVALID_PARAMETERS
One of the input parameters is invalid
DUPLICATE_NQRNAME
An attempt has been made to create a deep enqmodel nesting, or another enqmodel with the same nqrname is already installed.
DUPLICATE_ENABLED
An attempt to create an enabled enqmodel failed, because a less specific enqmodel is enabled.
CATALOG_WRITE_FAILED
COMMIT was specified but the record was not written to the catalogue.
GETMAIN_FAILED
The getmain for the NQRN storage failed.
DIRECTORY_ADD_FAILED
The DFHDDDIM ADD_ENTRY failed to add the ENQMODEL entry.
DIRECTORY_DELETE_FAILED
The DFHDDDIM DELETE_ENTRY failed to delete the ENQMODEL entry.
ACQUIRE_LOCK_FAILED
Attempt to acquire an exclusive NQRNAME lock failed.
RELEASE_LOCK_FAILED
Attempt to release an exclusive NQRNAME lock failed.

NQRN gate, DISCARD_ENQMODEL function

Remove an enqmodel definition from both the NQRN directory and from the NQRNAME_LIST.

If the enqmodel is not installed, an 'ENQMODEL_NOT_FOUND' exception is returned.

The ENQMODEL is put into the WAITING state until there are no enqueues in the local system which match the ENQNAME pattern. It is then removed from the local system.

Input parameters:

ENQMODEL The 8-character identifier of the resource to be DELETED.

Output parameters:

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER
[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:
ENQMODEL_NOT_FOUND
The name is not in the NQRN directory.
CATALOG_DELETE_FAILED
An attempt to delete the ENQMODEL ENTRY from the GCD failed.
ACQUIRE_LOCK_FAILED
Attempt to acquire an exclusive NQRNAME lock failed.

RELEASE_LOCK_FAILED
Attempt to release an exclusive NQRNAME lock failed.

NQRN gate, INQUIRE_ENQMODEL function

Uses directory DDLO_LOCATE to retrieve information about a specified enqmodel definition in the NQRN directory.

If found, it returns the 1 to 255 character NQRNAME, the 4-character SCOPE name, the enqmodel STATE and ann OK RESPONSE. Otherwise it returns an EXCEPTION REASON(ENQMODEL_NOT_FOUND).

Input parameters:

ENQMODEL The 8-character identifier of the entry to be returned.

Output parameters:

NQRNAME A buffer returning the 1 to 255 character name and length of the ENQ name or generic stem*

SCOPE Returns the 4-character scope identifier for the resource. Four blanks indicates that the enqueue has local scope.

STATE

ENABLED
Matching ENQ/DEQ requests should be processed.

DISABLED
Matching ENQ/DEQ requests should be rejected, and the issuing task abended abcode ENQ_DISABLED.

WAITING
Matching ENQ/DEQ requests should be rejected, and the issuing task abended abcode ENQ_DISABLED. There are INSTALL, CREATE, or DISCARD requests waiting to be processed.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

ENQMODEL_NOT_FOUND
The name does not exist in the directory.

DIRECTORY_LOCATE_FAILED
Directory DDLO_LOCATE failed with something other than NOT_FOUND.

ACQUIRE_LOCK_FAILED
Attempt to acquire a shared NQRNAME lock failed.

RELEASE_LOCK_FAILED
Attempt to release a shared NQRNAME lock failed.

NQRN gate, SET_ENQMODEL function

This function uses directory DDLO_LOCATE to see if an enqmodel entry exists in the NQRN directory. If found, it calls SET_ENQMODEL to enable or disable the entry. Otherwise it returns an EXCEPTION REASON(ENQMODEL_NOT_FOUND).

Enqmodels forming nested generic nqrnames must be enabled in order, from the most to the least specific. I.e. A more specific enqmodel may not be enabled if a less specific enqmodel is enabled. If attempted, msg NQ0107 is issued and EXCEPTION 'DUPLICATE_ENABLED' is returned to the caller.

You cannot enable/disable an enqmodel which is in the waiting state. If attempted, EXCEPTION 'ENQMODEL_WAITING' is returned to the caller.

Input parameters:

ENQMODEL The 8-character identifier of the entry to be enabled/disabled.

STATE

ENABLED
The enqmodel is to be enabled.
DISABLED
The enqmodel is to be disabled.

Output parameters:

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

ENQMODEL_NOT_FOUND
The name does not exist in the directory.
ENQMODEL_WAITING
The enqmodel is in the WAITING state.
DUPLICATE_ENABLED
Attempt to enable/disable an enqmodel failed, because a less specific enqmodel is enabled.
DIRECTORY_LOCATE_FAILED
A DDLO_LOCATE failed with something other than NOT_FOUND.
CATALOG_UPDATE_FAILED
Attempt to update the enqmodel on the global catalog failed.
ACQUIRE_LOCK_FAILED
Attempt to acquire an exclusive NQRNAME lock failed.
RELEASE_LOCK_FAILED
Attempt to release an exclusive NQRNAME lock failed.

NQIE gate, INTERPRET_ENQUEUE function

This function interprets the passed enqueue prior to it being returned by the EXEC CICS INQUIRE UOWENQ command. The function takes the enqueue to be interpreted as input and returns ENQUEUE_TYPE, RESOURCE and QUALIFIER to the caller (EXEC layer).

Each enqueue pool can either

- not have an interpreter and consequently not have its enqueues returned by the INQUIRE UOWENQ command
- rely upon a default interpreter supplied by NQ domain, (DFHNQIE)
- supply its own interpreter routine.

This is specified when the pool is created.

Input parameters:

POOL_NAME Name of the pool containing the enqueue to be interpreted.

Note that an interpreter may interpret enqueues from more than one pool.

POOL_TOKEN Token corresponding to the pool containing the enqueue to be interpreted

ENQUEUE_NAME A block (addr,len) identifying the full name of the enqueue to be interpreted.

ENQUEUE_NAME2_LENGTH
The length of the second part of the enqueue name if the enqueue was originally specified in two parts (i.e. ENQUEUE_NAME1 and ENQUEUE_NAME2).

If the ENQUEUE_NAME2 parameter wasn't originally specified for this enqueue then this will contain zero.

Output parameters:

RESOURCE_BUFFER
A buffer into which the data for the RESOURCE field is returned. The caller specifies the address and maximum length of the data area into which the RESOURCE data will be returned. If the data is too big for the buffer then the data is truncated and an OK response is returned. The actual length of the name is returned in resource_buffer_n.

QUALIFIER_BUFFER
A buffer into which the data for the QUALIFIER field is returned. The caller specifies the address and maximum length of the data area into which the QUALIFIER data will be returned. If the data is too big for the buffer then the data is truncated and an OK response is returned. The actual length of the name is returned in qualifer_buffer_n.

If there is no QUALIFIER data then no data should be returned and the length of the data (qualifier_buffer_n) be returned as zero.

ENQUEUE_TYPE
The TYPE of the enqueue being returned.

The possible values are as follows and these map onto the CVDA values for the TYPE field as detailed under the EXEC CICS INQUIRE UOWENQ command.
- DATASET
- EXECENQ
- EXECENQADDR
- FILE
- TDQUEUE
- TSQUEUE

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_ENQUEUE

Enqueue domain's generic gates

Table 75 summarizes the NQ domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 75. NQ domain's generic gates

Gate	Trace	Function	Format
DMDM	NQ 0101 NQ 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	NQ 0501 NQ 0502	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
RMRO	NQ 0201 NQ 0202	PERFORM_PREPARE PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT	RMRO

For descriptions of these functions and their input and output parameters, refer to the Sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—Chapter 78, "Domain manager domain (DM)," on page 663

Format STST—"System programming command flows" on page 264

Format RMRO—Chapter 99, "Recovery Manager Domain (RM)," on page 1061

PERFORM_PREPARE is a no-op. PERFORM_COMMIT releases enqueues. PERFORM_SHUNT make active enqueues retained. PERFORM_UNSHUNT makes retained enquires active.

The Domain Manager gates perform normal internal state initialisation and termination functions.

Modules

Module	Function
DFHNQDM	Handles the following requests: INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHNQDUF	Formats the NQ domain control blocks in a CICS system.
DFHNQNQ	Handles the following requests: CREATE_ENQUEUE_POOL REACQUIRE_ENQUEUE DEACTIVATE SET_NQRNAME_LIST DEQUEUE_TASK
DFHNQED	Handles the following requests: ENQUEUE DEQUEUE
DFHNQEDI	Inline version of DFHNQED.
DFHNQIB	Handles the following requests: INQUIRE_ENQUEUE START_BROWSE_ENQUEUE GET_NEXT_ENQUEUE END_BROWSE_ENQUEUE
DFHNQRN	Handles the following requests: INQUIRE_NQRNAME ADD_REPLACE_ENQMODEL DISCARD_ENQMODEL REMOVE_ENQMODEL INQUIRE_ENQMODEL START_BROWSE_ENQMODEL GET_NEXT_ENQMODEL END_BROWSE_ENQMODEL SET_ENQMODEL COMMIT_ENQMODEL RESTORE_DIRECTORY
DFHNQIE	Handles the following requests: INTERPRET_ENQUEUE
DFHNQST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHNQTRI	Provides a trace interpretation routine for CICS dumps and traces.

Exits

The XNQEREQ and XNQEREQC global user exit points are invoked respectivly before and after each EXEC ENQ or DEQ request to the NQ domain.

Trace

The point IDs for the NQ domain are of the form NQ xxxx; the corresponding trace levels are NQ 1, NQ 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 93. Object Transaction Service domain (OT)

The object transaction service domain provides services to manage OTS transactions.

Object Transaction Service domain's specific gates

Table 76 summarizes the OT domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 76. Object Transaction Service domain's specific gates

Gate	Trace	Function	XPI
OTTR	OT 0401	IMPORT_TRAN	NO
	OT 0402	BEGIN_TRAN	NO
		COMMIT_ONE_PHASE	NO
		PREPARE	NO
		COMMIT	NO
		ROLLBACK	NO
		SET_ROLLBACK_ONLY	NO
OTSU	XM 0501	ADD_SUBORDINATE	NO
	XM 0502	SET_VOTE	NO
		FORGET	NO
		RESYNC	
OTCO	XM 0601	SET_COORDINATOR	NO
	XM 0602	FORGET	NO
		RESYNC	NO
OTRM	XM 0701	PERFORM_PRELOGGING	NO
	XM 0702	PERFORM_PREPARE	NO
		PERFORM_COMMIT	NO
		REPLY_DO_COMMIT	NO
		SEND_DO_COMMIT	NO
		PERFORM_SHUNT	NO
		PERFORM_UNSHUNT	NO
		START_BACKOUT	NO
		DELIVER_BACKOUT_DATA	NO
		END_BACKOUT	NO

OTTR gate, IMPORT_TRAN function

The IMPORT_TRAN function of the OTTR gate is used to import an OTS transaction to a task.

Input parameters

FORMAT_ID The OTS transactions format identifier.

BQUAL_LEN The batch qualifer length of the OTS transaction.

TID_BLOCK_IN The OTS transaction identifier (TID) of the transaction being imported.

TIMEOUT The OTS transaction timeout value.

LOGICAL_SERVER
The name of the logical server within which the transaction is executing.

PUBLIC_ID The Request Stream public identifier associated with the transaction.

Output parameters

UOW_ID The identifier of the logical unit of work into which the OTS transaction was imported.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TID_TOO_LONG
INVALID	INVALID_FUNCTION

OTTR gate, BEGIN_TRAN function

The BEGIN_TRAN function of the OTTR gate is used to create a new OTS transaction.

Input parameters

[TIMEOUT] The timeout specified for the new OTS transaction.

TID_BUFFER_OUT
The OTS transaction identifier (TID) of the transaction created.

LOGICAL_SERVER
The name of the logical server within which the transaction is executing.

PUBLIC_ID The Request Stream public identifier associated with the transaction.

Output parameters

FORMAT_ID The OTS transactions format identifier.

BQUAL_LEN The batch qualifer length of the OTS transaction.

[TIMEOUT] The default OTS transaction timeout value.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TID_TOO_LONG UOW_ROLLEDBACK

OTTR gate, COMMIT_ONE_PHASE function

The COMMIT_ONE_PHASE function of the OTTR gate is used to attempt to commit the current OTS transaction.

Input parameters

None

Output parameters

STATUS The outcome of the OTS transaction. It can have either of these values:
COMMITTED|ROLLEDBACK

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None

OTTR gate, PREPARE function

The PREPARE function of the OTTR gate is used to perform the first phase of the syncpoint of an OTS transaction.

Input parameters

None

Output parameters

VOTE The vote from first phase of syncpoint. It can have any of these values:
YES|NO|READ_ONLY|HEURISTIC_MIXED

RESPONSE is the domain's response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None

OTTR gate, COMMIT function

The COMMIT function of the OTTR gate is used to perform the second phase of the syncpoint of an OTS transaction.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UOW_ROLLEDBACK

OTTR gate, SET_ROLLBACK_ONLY function

The SET_ROLLBACK_ONLY function of the OTTR gate is used to ensure that the OTS transaction will rollback when it comes to syncpoint.

Input parameters

None.

Output parameters

RESPONSE
is the domain|s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None

OTSU gate, ADD_SUBORDINATE function

The ADD_SUBORDINATE function of the OTSU gate is used add a subordinate participant to the OTS transaction.

Input parameters

IOR_BLOCK
Block containing the CORBA IOR of the OTS Resource that is being added as a subordinate participant in the OTS transaction.

HOST_BLOCK
Block containing the name of the TCPIP host where the subordinate OTS resource resides.

Output parameters

SUBORDINATE_TOKEN
token representing the added Resource.

RESPONSE
is the domain|s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IOR_TOO_LONG HOST_TOO_LONG
DISASTER	ADD_LINK_FAILED

OTSU gate, SET_VOTE function

The SET_VOTE function of the OTSU gate is used record the vote that results from a PREPARE method being invoked on the OTS Resource represented by the given SUBORDINATE_TOKEN.

Input parameters

SUBORDINATE_TOKEN
Token representing the subordinate OTS resource.

VOTE
The vote resulting from the first phase of syncpoint on the subordinate resource. It can have one of the following values:

YES|NO|READ_ONLY|HEURISTIC_MIXED|
HEURISTIC_COMMIT|HEURISTIC_ROLLBACK|HEURISTIC_HAZARD

Output parameters

RESPONSE
is the domain|s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_SUBORDINATE INVALID_VOTE
DISASTER	RECORD_VOTE_FAILED

OTSU gate, FORGET function

The FORGET function of the OTSU gate is used signal the fact that the obligation to the subordinate resource has been discharged.

Input parameters

SUBORDINATE_TOKEN
Token representing the subordinate OTS resource.

Output parameters

RESPONSE
is the domain|s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_SUBORDINATE
DISASTER	INBOUND_FLOW_FAILED

OTSU gate, RESYNC function

The RESYNC function of the OTSU gate is used to initiate the resynchronisation protocol with the subordinate resource identified by the given IOR.

Input parameters

IOR_BLOCK
Block containing the CORBA IOR of the OTS Resource with which to resynchronise.

UOWID
identification of the local logical unit of work managing the OTS transaction.

Output parameters

RESPONSE
is the domain|s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None

OTCO gate, SET_COORDINATOR function

The SET_COORDINATOR function of the OTCO gate is used to make known the CORBA object that will coordinate this part of the OTS transaction.

Input parameters

IOR_BLOCK
Block containing the CORBA IOR of the OTS Coordinator.

HOST_BLOCK
Block containing the name of the TCPIP host where the coordinator resides.

Output parameters

COORDINATOR_TOKEN
token representing the coordinator.

RESPONSE is the domain|s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IOR_TOO_LONG HOST_TO_LONG

OTCO gate, FORGET function

The FORGET function of the OTCO gate is used signal the fact that the obligation to the coordinator has been discharged.

Input parameters

SUBORDINATE_TOKEN
Token representing the subordinate OTS resource.

Output parameters

RESPONSE is the domain|s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_SUBORDINATE
DISASTER	INBOUND_FLOW_FAILED

Modules

Module	Function
DFHOTDM	Domain initialisation and termination. PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHOTRM	Handles the following requests: ATTACH
DFHOTTR	Handles requests on the OTTR gate.
DFHOTSU	Handles requests on the OTSU gate.
DFHOTCO	Handles requests on the OTCO gate.
DFHOTDUF	OT domain offline dump formatting routine
DFHOTTRI	Interprets OT domain trace entries

Exits

None

Trace

The point IDs for the OT domain are of the form OTxxxx; the corresponding trace levels are OT 1, OT 2 and Exc.

Object Transaction Service domain (OT)

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 94. Parameter manager domain (PA)

The parameter manager domain (also sometimes known simply as "parameter manager") provides a facility to inform CICS domains of system parameters during CICS initialization. These **system initialization parameters** are specified in the system initialization table (SIT), and as temporary override parameters read from the SYSIN data stream or specified interactively at the system console.

The parameter manager domain also provides an operator correction facility for incorrectly specified system initialization parameter keywords early in CICS initialization. To use this facility, the user must specify the PARMERR system initialization parameter.

Parameter manager domain's specific gate

Table 77 summarizes the parameter manager domain's specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 77. Parameter manager domain's specific gate

Gate	Trace	Function	XPI
PAGP	PA 0101	FORCE_START	NO
	PA 0102	GET_PARAMETERS	NO
		INQUIRE_START	NO

PAGP gate, FORCE_START function

The FORCE_START function of the PAGP gate is used to override the type of start requested by the START system initialization parameter. It is currently used to force START=AUTO if the MVS automatic restart manager indicates that CICS is being automatically restarted with the original startup JCL (so that CICS does not get a COLD start that the original JCL might have asked for).

Input parameters

START_TYPE
specifies the type of CICS start to be forced. It can have either of these values:

COLD|AUTO

Output parameters

RESPONSE
is the parameter manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_POSSIBLE

PAGP gate, GET_PARAMETERS function

The GET_PARAMETERS function of the PAGP gate is used to get the initialization parameters for a requesting domain.

Input parameters

FORCE_ALL
specifies whether all parameters are required, even on a non-cold start. It can have either of these values:

YES|NO

Output parameters

PARAMETERS_TRANSFERRED
: indicates to the calling domain whether any system parameters were transferred successfully by the parameter manager domain. It can have either of these values:

  YES|NO

RESPONSE
: is the parameter manager domain's response to the call. It can have any one of these values:

  OK|EXCEPTION|DISASTER

PAGP gate, INQUIRE_START function

The INQUIRE_START function of the PAGP gate is used to find out the type of start that CICS is to perform. This information is used to determine whether domains need to perform a cold or warm start.

Input parameters

None.

Output parameters

START
: specifies the type of start CICS is to perform. It can have any one of these values:

  COLD|WARM|LOGTERM

RESPONSE
: is the parameter manager domain's response to the call. It can have any one of these values:

  OK|EXCEPTION|DISASTER

Parameter manager domain's generic gate

Table 78 summarizes the parameter manager domain's generic gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 78. Parameter manager domain's generic gate

Gate	Trace	Function	Format
DMDM	PA 0201 PA 0202	PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

You can find descriptions of these functions and their input and output parameters, in the section dealing with the corresponding generic format, in "Domain manager domain's generic formats" on page 669.

In preinitialization processing, the parameter manager domain reads system initialization (override) parameters from the startup job stream and, if requested, from the SYSIN data set and the console.

If a system initialization table (SIT) has been specified, that is loaded into storage. Otherwise, the default SIT is loaded. The override parameters are applied to the SIT, and related parameters are checked for consistency. Errors are reported, but no action is taken.

The parameter manager domain also provides services to other domains as they preinitialize. It informs them of the type of start (cold or auto), and supplies information as required from the SIT.

In initialization processing, the parameter manager domain waits for all the other domains to complete their initialization, and then writes a warm start record to the catalog.

The parameter manager domain does no quiesce processing or termination processing.

Modules

Module	Function
DFHPADM	Parameter manager domain initialization and termination
DFHPADUF	An offline routine to format system dump information
DFHPAGP	Passes initialization parameters to domains requesting GET_PARAMETERS
DFHPAIO	Communicates with the SYSIN data set and operator console
DFHPASY	System initialization override parameter checker and syntax parser
DFHPATRI	An offline routine to format trace points

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the parameter manager domain are of the form PA xxxx; the corresponding trace levels are PA 1, PA 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 95. Program manager domain (PG)

The program manager domain provides support for the following areas of CICS:

- Program control functions; EXEC CICS LINK, XCTL, LOAD, RELEASE, and RETURN
- Transaction ABEND and condition handling functions; EXEC CICS ABEND, HANDLE ABEND, HANDLE CONDITION and HANDLE AID
- Related functions such as invoking user-replaceable programs, global user exits, and task-related user exits
- Autoinstall for programs, mapsets, and partitionsets.

Program manager domain's specific gates

Table 79 summarizes the program manager domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 79. Program manager domain's specific gates

Gate	Trace	Function	XPI
PGAI	PG 0E01 PG 0E02	Kernel subroutine called internally from program manager	NO NO
PGAQ	PG 0401 PG 0402	INQUIRE_AUTOINSTALL SET_AUTOINSTALL	YES YES
PGCH	PG 1700 PG 1701	BIND_CHANNEL COPY_CHANNEL CREATE_CHANNEL DELETE_CHANNEL DELETE_OWNED_CHANNELS DETACH_CHANNEL INQUIRE_BOUND_CHANNEL INQUIRE_CHANNEL INQUIRE_CHANNEL_BY_TOKEN INQUIRE_CURRENT_CHANNEL RENAME_CHANNEL SET_CURRENT_CHANNEL	NO NO NO NO NO NO NO NO NO NO NO NO
PGCP	PG 1800 PG 1801	COPY_CONTAINER_POOL CREATE_CONTAINER_POOL DELETE_CONTAINER_POOL, INQUIRE_CONTAINER_POOL	NO NO NO NO
PGCR	PG 1900 PG 1901	COPY_CONTAINER DELETE_CONTAINER ENDBR_CONTAINER GET_CONTAINER_INTO GET_CONTAINER_LENGTH GET_CONTAINER_SET GETNEXT_CONTAINER INQUIRE_BROWSE_CONTEXT INQUIRE_CONTAINER INQUIRE_CONTAINER_BY_TOKEN MOVE_CONTAINER PUT_CONTAINER SET_CONTAINER TRACE_CONTAINERS	NO NO NO NO NO NO NO NO NO NO NO NO NO NO
PGDD	PG 0301 PG 0302	DEFINE_PROGRAM DELETE_PROGRAM	NO NO
PGEX	PG 0C01 PG 0C02	INITIALIZE_EXIT TERMINATE_EXIT	NO NO

Table 79. Program manager domain's specific gates (continued)

Gate	Trace	Function	XPI
PGHM	PG 0700 PG 0701	SET_CONDITIONS IGNORE_CONDITIONS INQ_CONDITION SET_AIDS INQ_AID SET_ABEND INQ_ABEND PUSH_HANDLE POP_HANDLE FREE_HANDLE_TABLES CLEAR_LABELS	NO NO
PGIS	PG 0500 PG 0501	INQUIRE_PROGRAM INQUIRE_CURRENT_PROGRAM SET_PROGRAM START_BROWSE_PROGRAM GET_NEXT_PROGRAM END_BROWSE_PROGRAM REFRESH_PROGRAM	YES YES YES YES YES YES NO
PGLD	PG 0601 PG 0602	LOAD_EXEC LOAD RELEASE_EXEC RELEASE	NO NO NO NO
PGLE	PG 1101 PG 1102	LINK_EXEC	NO
PGLK	PG 0B01 PG 0B02	LINK LINK_PLT	NO NO
PGLU	PG 0A01 PG 0A02	LINK_URM	NO NO
PGPG	PG 0901 PG 0902	INITIAL_LINK	NO NO
PGRE	PG 1201 PG 1202	PREPARE_RETURN_EXEC	NO
PGXE	PG 1301 PG 1302	PREPARE_XCTL_EXEC	NO NO
PGXM	PG 0901 PG 0902	INITIALIZE_TRANSACTION TERMINATE_TRANSACTION	NO NO

Note: PGRE is only called for EXEC RETURN statements which have input parameters (COMMAREA, INPUTMSG, or TRANSID) specified. If no input parameters are specified, there is no trace of PGRE after the EIP trace of the RETURN statement.

PGAQ gate, INQUIRE_AUTOINSTALL function

The INQUIRE_AUTOINSTALL function of the PGAQ gate is used to inquire about attributes of the program autoinstall function.

Input parameters

None.

Output parameters

[AUTOINSTALL_STATE]
is the state of the program autoinstall function. It can have either of these values:

ACTIVE|INACTIVE

[AUTOINSTALL_CATALOG]
identifies if program autoinstall events are cataloged. It can have any of these values:

MODIFY|NONE|ALL

[AUTOINSTALL_EXIT_NAME]
is the name of the program autoinstall exit program.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION

PGAQ gate, SET_AUTOINSTALL function

The SET_AUTOINSTALL function of the PGAQ gate is used to set attributes of the program autoinstall function.

Input parameters

[AUTOINSTALL_STATE]
is the state of the program autoinstall function. It can have either of these values:

ACTIVE|INACTIVE

[AUTOINSTALL_CATALOG]
identifies if program autoinstall events are cataloged. It can have any of these values:

MODIFY|NONE|ALL

[AUTOINSTALL_EXIT_NAME]
is the name of the program autoinstall exit program.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION

PGCH gate, BIND_CHANNEL function

The BIND_CHANNEL function of the PGCH gate is used to make the specified channel the channel used on the initial link.

Input parameters

CHANNEL_TOKEN
is a token referencing the channel to be used on the initial link.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_ALREADY_SET
INVALID	INVALID_LINK_LEVEL INVALID_TOKEN

CHANNEL_ALREADY_SET
Only one bind can occur for each task.

INVALID_LINK_LEVEL
The command was executed outside a program manager environment.

INVALID_TOKEN
The specified token does not address a channel control block.

PGCH gate, COPY_CHANNEL function

The COPY_CHANNEL function of the PGCH gate is used to take a copy of a channel and all its containers. The copy has the same name as the original, but is not on any chain. This function is required by the START command.

Input parameters

CHANNEL_TOKEN
is a token referencing the channel to be copied.

Output parameters

COPIED_CHANNEL_TOKEN
A token referencing a copy of the specified channel (used on START and RETURN commands).

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_TOKEN

INVALID_TOKEN
The specified token does not address a channel control block.

PGCH gate, CREATE_CHANNEL function

The CREATE_CHANNEL function of the PGCH gate is used to create a channel.

Input parameters

[CCSID]
is the default coded character set identifier (CCSID) for character data in this channel.

CHANNEL_NAME
is the 16-character name of the channel to be created.

[CURRENT_CHANNEL]
whether or not the created channel is to be the current channel of the current link level. It can have either of these values:

YES|NO

[LINK_LEVEL]
whether the channel is to be created on the current chain, the previous link level's chain, or on no chain (NONE). LINK_LEVEL can have any of these values:

CURRENT|PREVIOUS|NONE

NONE is used when creating a channel for transfer on a START or RETURN command.

Output parameters

[CHANNEL_TOKEN]
is a token referencing the newly-created channel.

[CONTAINER_POOL_TOKEN]
is a token to access a pool of containers.

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_ALREADY_EXISTS CHANNEL_ALREADY_SET
INVALID	CCSID_INVALID INVALID_CHANNEL_NAME INVALID_LINK_LEVEL INVALID_PARAMETERS INVALID_TOKEN

CCSID_INVALID
The specified CCSID is incorrect.

CHANNEL_ALREADY_EXISTS
A channel with this name already exists.

CHANNEL_ALREADY_SET
Only one current channel is possible.

INVALID_CHANNEL_NAME
The channel name contains invalid characters.

INVALID_LINK_LEVEL
A link-level of PREVIOUS was specified in a top-level program, or the command is executed outside a program manager environment.

INVALID_PARAMETERS
You have specified an invalid combination of parameters.

INVALID_TOKEN
The specified token does not address a channel control block.

PGCH gate, DELETE_CHANNEL function

The DELETE_CHANNEL function of the PGCH gate is used to delete a channel. This command can be used to delete channels when they are bound to principal facilities, but not to PLCBs.

Input parameters

CHANNEL_TOKEN
is a token referencing the channel to be deleted.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_ATTACHED
INVALID	INVALID_TOKEN

CHANNEL_ATTACHED
The channel is attached to PLCB chain and cannot be deleted.

INVALID_TOKEN
The specified token does not address a channel control block.

PGCH gate, DELETE_OWNED_CHANNELS function

The DELETE_OWNED_CHANNELS function of the PGCH gate is used to delete all channels from the channel chain. If the current channel is owned by this link level, it is deleted as well. The container pool associated with each channel is also deleted. This ends any browse in progress and deletes all containers.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_LINK_LEVEL

INVALID_LINK_LEVEL
The command was executed outside a program manager environment.

PGCH gate, DETACH_CHANNEL function

The DETACH_CHANNEL function of the PGCH gate is used to detach a channel. The channel may be the current channel, or on the PLCB chain. The channel's containers are only deleted if DELETE(YES) is specified. It is implied that a SET_CURRENT_CHANNEL will be done with this channel at some time.

A channel can only be detached from the current link level.

Input parameters

CHANNEL_TOKEN
is a token referencing the channel to be detached.

[DELETE]
whether the channel's containers should be deleted. DELETE can have either of these values:

YES|NO

[FREE_SET_STORAGE]
whether the channel's storage should be freed. FREE_SET_STORAGE can have either of these values:

YES|NO

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_NOT_FOUND
INVALID	INVALID_TOKEN INVALID_LINK_LEVEL

CHANNEL_NOT_FOUND
The channel was not found on the PLCB channel chain.

INVALID_LINK_LEVEL
The command was executed outside a program manager environment.

INVALID_TOKEN
The specified token does not address a channel control block.

PGCH gate, INQUIRE_BOUND_CHANNEL function

The INQUIRE_BOUND_CHANNEL function of the PGCH gate is used to get information about the channel that is bound to the current transaction. This may or may not be the current channel. This request may be issued outside a program manager environment.

Output parameters

[CHANNEL_TOKEN]
is a token referencing the bound channel.

[CHANNEL_NAME]
is the name of the bound channel.

[CONTAINER_POOL_TOKEN]
is a token referencing the container pool of the bound channel.

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_NOT_FOUND

CHANNEL_NOT_FOUND
There is no bound channel.

PGCH gate, INQUIRE_CHANNEL function

The INQUIRE_CHANNEL function of the PGCH gate is used to retrieve the properties of a named channel, including its address (returned as a token). To find the named channel, CICS scans the channels accessible from the specified link level.

Input parameters

CHANNEL_NAME
is the name of the channel to be to be enquired upon.

[LINK_LEVEL]
whether the named channel is on the current chain or the previous link level's chain. LINK_LEVEL can have either of these values:

`CURRENT|PREVIOUS`

Output parameters

[CCSID]
is the default coded character set identifier (CCSID) for character data in the named channel.

[CHANNEL_TOKEN]
is a token referencing the named channel.

[CONTAINER_POOL_TOKEN]
is a token referencing the container pool of the named channel.

[CURRENT_CHANNEL]
whether the named channel is the current channel. CURRENT_CHANNEL can have either of these values:

YES|NO

[OWNER]
whether the named channel is owned by the specified link level. OWNER can have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_NOT_FOUND
INVALID	INVALID_LINK_LEVEL

CHANNEL_NOT_FOUND
No channel can be found at the specified link level.

INVALID_LINK_LEVEL
A link-level of PREVIOUS was specified in a top-level program, or the command was executed outside a program manager environment.

PGCH gate, INQUIRE_CHANNEL_BY_TOKEN function

The INQUIRE_CHANNEL_BY_TOKEN function is used to retrieve the properties of a channel (which is specified by token).

Input parameters

CHANNEL_TOKEN
is a token referencing the channel to be enquired upon.

Output parameters

[CCSID]
is the default coded character set identifier (CCSID) for character data in the specified channel.

[CHANNEL_NAME]
is the name of the specified channel.

[CONTAINER_POOL_TOKEN]
is a token referencing the container pool of the specified channel.

[CURRENT_CHANNEL]
whether the specified channel is the current channel. CURRENT_CHANNEL can have either of these values:

YES|NO

[OWNER]
whether the specified channel is owned by the specified link level. OWNER can have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_TOKEN
INVALID	INVALID_LINK_LEVEL

CHANNEL_NOT_FOUND
No channel can be found at the specified link level.

INVALID_LINK_LEVEL
A link-level of PREVIOUS was specified in a top-level program, or the command was executed outside a program manager environment.

INVALID_TOKEN
The specified token does not address a channel control block.

PGCH gate, INQUIRE_CURRENT_CHANNEL function

The INQUIRE_CURRENT_CHANNEL function of the PGCH gate is used to retrieve the properties of the current channel.

Output parameters

[CCSID]
is the default coded character set identifier (CCSID) for character data in the current channel.

[CHANNEL_NAME]
is the name of the current channel.

[CHANNEL_TOKEN]
is a token referencing the current channel.

[CONTAINER_POOL_TOKEN]
is a token referencing the container pool of the current channel.

[OWNER]
whether the current channel is owned by the specified link level. OWNER can have either of these values:

`YES|NO`

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_NOT_FOUND
INVALID	INVALID_LINK_LEVEL

CHANNEL_NOT_FOUND
The current link level was not passed a channel.

INVALID_LINK_LEVEL
The command was executed outside a program manager environment.

PGCH gate, RENAME_CHANNEL function

The RENAME_CHANNEL function of the PGCH gate is used to rename a channel.

Input parameters

CHANNEL_NAME
is the new name of the channel, after it has been renamed.

CHANNEL_TOKEN
is a token referencing the channel to be renamed. The token must have been obtained using the INQUIRE_CHANNEL command.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_ALREADY_EXISTS INVALID_CHANNEL_NAME INVALID_TOKEN

CHANNEL_ALREADY_EXISTS
A channel with the specified channel name already exists.

INVALID_CHANNEL_NAME
The new channel name contains invalid characters.

INVALID_TOKEN
The specified channel token does not address a channel control block.

PGCH gate, SET_CURRENT_CHANNEL function

The SET_CURRENT_CHANNEL function of the PGCH gate is used to make the specified channel the current channel for the current link level.

If OWNER(YES) is specified, the channel is added to the current link level's chain.

Input parameters

CHANNEL_TOKEN
is a token referencing the channel that is to become the current channel for this link level.

[OWNER]
whether the specified channel is owned by the current link level. If OWNER(YES) is specified, the channel is added to the current link level's chain. OWNER can have either of these values:

YES|NO

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CHANNEL_ALREADY_EXISTS INVALID_TOKEN
INVALID	INVALID_LINK_LEVEL

CHANNEL_ALREADY_EXISTS
A channel with the same name as the specified channel already exists on the current link level's chain.

INVALID_LINK_LEVEL
The command was executed outside a program manager environment.

INVALID_TOKEN
The specified channel token does not address a channel control block.

PGCP gate, COPY_CONTAINER_POOL function

The COPY_CONTAINER_POOL function of the PGCP gate is used to copy all the containers in a container pool to another container pool.

Input parameters

POOL_TOKEN
is a token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

Output parameters

COPIED_POOL_TOKEN
is a token that maps to the pool to which all containers have been copied from the pool referenced by POOL_TOKEN.

RESPONSE
is the domain's response to the call. It can have any of these values:
`OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_POOL_TOKEN

PGCP gate, CREATE_CONTAINER_POOL function

The CREATE_CONTAINER_POOL function of the PGCP gate is used to create a container pool.

Input parameters

[CCSID]
is the default coded character set identifier of the character data in the pool to be created.

Output parameters

POOL_TOKEN
is a token that references the container pool that has been created.

RESPONSE
is the domain's response to the call. It can have any of these values:
`OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_POOL_TOKEN

PGCP gate, DELETE_CONTAINER_POOL function

The DELETE_CONTAINER_POOL function of the PGCP gate is used to delete a container pool.

Input parameters

POOL_TOKEN
is a token that identifies the container pool to be deleted.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_POOL_TOKEN

PGCP gate, INQUIRE_CONTAINER_POOL function

The INQUIRE_CONTAINER_POOL function of the PGCP gate is used to inquire about the attributes of a container pool.

Input parameters

POOL_TOKEN
is a token that identifies the container pool to be inquired upon.

Output parameters

[CCSID]
is the default coded character set identifier (CCSID) for character data in the containers in the pool.

[NUMBER_OF_CONTAINERS]
is the number of containers that the pool contains.

[POOL_SIZE]
is the size, in bytes, of the data in the pool.

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_POOL_TOKEN

PGCR gate, COPY_CONTAINER function

The COPY_CONTAINER function of the PGCR gate is used to copy a container from one container pool to another. Both pools must already have been created.

Input parameters

[AS_CONTAINER_NAME]
is the name by which the copied container is to be known in the target container pool.

[CONTAINER_NAME]
is the name of the container to be copied.

[CONTAINER_TOKEN]
is a token referencing the container to be copied.

[POOL_TOKEN]
is a token referencing the source container pool (that is, the pool from which the container is to be copied).

[TO_POOL_TOKEN]
is a token referencing the target container pool (that is, the pool to which the container is to be copied).

[TYPE]
whether the container is visible only to CICS, or to user programs as well. TYPE can have either of these values:

`CICS|USER`

Output parameters

[CONTAINER_TOKEN_OUT]
is a token representing the new copy of the container.

[GENERATION_NUMBER]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

[INITIAL_GENERATION]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

REASON
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CONTAINER_NOT_FOUND INVALID_AS_CONTAINER_NAME
INVALID	INVALID_CONTAINER_TOKEN INVALID_POOL_TOKEN INVALID_PARAMETERS INVALID_TO_POOL_TOKEN

PGCR gate, DELETE_CONTAINER function

The DELETE_CONTAINER function of the PGCR gate is used to delete a container and its data. The container is identified using its name, the container pool to which it belongs, and its type.

Input parameters

[CALLER]
is the call part of an API call. It can have either of these values:
EXEC|SYSTEM

[CONTAINER_NAME]
is the name of the container to be deleted.

[CONTAINER_TOKEN]
is a token representing the container to be deleted.

[POOL_TOKEN]
is a token referencing the container pool that contains the container to be deleted.

[TYPE]
whether the container is visible only to CICS, or to user programs as well. TYPE can have either of these values:
CICS|USER

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CONTAINER_NOT_FOUND READONLY_CONTAINER
INVALID	INVALID_CONTAINER_TOKEN INVALID_PARAMETERS INVALID_POOL_TOKEN

PGCR gate, ENDBR_CONTAINER function

The ENDBR_CONTAINER function of the PGCR gate is used to end a browse of containers.

Input parameters

BROWSE_TOKEN
is a browse token referencing the next container in the container pool being browsed.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN

PGCR gate, GET_CONTAINER_INTO function

The GET_CONTAINER function of the PGCR gate is used to get the data from a container into an area provided by the caller. The container is identified using a pool token, together with the container's name and type. Note that LENGTH_ERROR indicates that as much data as possible has been copied.

It is the user's responsibility that DATA_TOKEN_OUT is specified on the next call as DATA_TOKEN_IN. The first call doesn't have a DATA_TOKEN_IN.

Input parameters

[CALLER]
is the call part of an API call. It can have either of these values:

EXEC|SYSTEM

[CCSID]
if conversion is specified (see the CONVERT option), CCSID is the coded character set identifier to which the character data in the container should be converted.

[CONTAINER_NAME]
is the name of the container from which the data is to be obtained.

[CONTAINER_TOKEN]
is a token representing the container from which the data is to be obtained.

[CONVERT]
whether the data in the container should be converted. It can have either of these values:

YES|NO

The default value is YES.

[DATA_TOKEN_IN]
is a token referencing the data in the container.

The value returned in DATA_TOKEN_OUT on one GET_CONTAINER_INTO call must be specified on the next call as DATA_TOKEN_IN. (The first GET_CONTAINER_INTO call for this container doesn't have a DATA_TOKEN_IN.)

ITEM_BUFFER
On input, ITEM_BUFFER_P is a pointer to a receiving area of length ITEM_BUFFER_M. On output, the value ITEM_BUFFER_N is set to the actual length returned.

[POOL_TOKEN]
is a token referencing the container pool to which the container belongs.

[TYPE]
whether the container is visible only to CICS, or to user programs as well. TYPE can have either of these values:

CICS|USER

Output parameters

[CONTAINER_CCSID]
is the coded character set identifier of the extracted data.

[DATA_TOKEN_OUT]
is a token referencing the data in the container.

The value returned in DATA_TOKEN_OUT on one GET_CONTAINER_INTO call must be specified on the next call as DATA_TOKEN_IN. (The first GET_CONTAINER_INTO call for this container doesn't have a DATA_TOKEN_IN.)

[DATATYPE]
is the format of the data. It can have either of these values:

CHAR|BIT

[GENERATION_NUMBER]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

[INITIAL_GENERATION]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

[USERACCESS]
whether USER containers can be updated by API commands. USERACCESS can have either of these values:

READONLY|ANY

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CCSID_CONVERSION_ERROR CCSID_INVALID CCSID_PAIR_UNSUPPORTED CCSID_PARTIAL_CONVERSION CONTAINER_NOT_FOUND INVALID_DATA_TOKEN_IN LENGTH_ERROR MORE_DATA
INVALID	INVALID_CONTAINER_TOKEN INVALID_POOL_TOKEN INVALID_PARAMETERS

PGCR gate, GET_CONTAINER_LENGTH function

The GET_CONTAINER_LENGTH function of the PGCR gate is used to discover the length, in bytes, of the data in a container.

Input parameters

[CALLER]
is the call part of an API call. It can have either of these values:

EXEC|SYSTEM

[CCSID]
is the coded character set identifier to which the character data in the container should be converted.

[CONTAINER_NAME]
is the name of the container that holds the data.

[CONTAINER_TOKEN]
is a token representing the container that holds the data.

[POOL_TOKEN]
is a token referencing the container pool to which the container belongs.

[TYPE]
whether the container is visible only to CICS, or to user programs as well. TYPE can have either of these values:

`CICS|USER`

Output parameters

[CONTAINER_CCSID]
is the coded character set identifier of the character data in the container.

[DATA_LENGTH]
is the length, in bytes, of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the *converted* data.

[DATATYPE]
is the format of the data. It can have either of these values:

`CHAR|BIT`

[GENERATION_NUMBER]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

[INITIAL_GENERATION]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

[USERACCESS]
whether USER containers can be updated by API commands. USERACCESS can have either of these values:

`READONLY|ANY`

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

REASON
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CCSID_CONVERSION_ERROR CCSID_IGNORED CCSID_INVALID CCSID_PAIR_UNSUPPORTED CCSID_PARTIAL_CONVERSION CONTAINER_NOT_FOUND
INVALID	INVALID_CONTAINER_TOKEN INVALID_PARAMETERS INVALID_POOL_TOKEN

PGCR gate, GET_CONTAINER_SET function

The GET_CONTAINER_SET function of the PGCR gate is used to get the data from a container and copy it into an area provided by the CICS program domain. The container is identified using a pool token, together with its name and type.

Input parameters

[CALLER]
is the call part of an API call. It can have either of these values:

`EXEC|SYSTEM`

[CCSID]
: if conversion is specified (see the CONVERT option), CCSID is the coded character set identifier to which the character data in the container should be converted.

[CONTAINER_NAME]
: is the name of the container from which the data is to be obtained.

[CONTAINER_TOKEN]
: is a token representing the container from which the data is to be obtained.

[CONVERT]
: whether the data in the container should be converted. It can have either of these values:

`YES|NO`

The default value is YES.

[POOL_TOKEN]
: is a token referencing the container pool to which the container belongs.

[TYPE]
: whether the container is visible only to CICS, or to user programs as well. TYPE can have either of these values:

`CICS|USER`

Output parameters

[CONTAINER_CCSID]
: is the coded character set identifier of the extracted data.

[DATATYPE]
: is the format of the data. It can have either of these values:

`CHAR|BIT`

[GENERATION_NUMBER]
: Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

[INITIAL_GENERATION]
: Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

ITEM_DATA
: The address and length of the SET storage returned.

[USERACCESS]
: whether USER containers can be updated by API commands. USERACCESS can have either of these values:

`READONLY|ANY`

RESPONSE
: is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

REASON
: is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CCSID_CONVERSION_ERROR CCSID_IGNORED CCSID_INVALID CCSID_PAIR_UNSUPPORTED CCSID_PARTIAL_CONVERSION CONTAINER_NOT_FOUND
INVALID	INVALID_CONTAINER_TOKEN INVALID_PARAMETERS INVALID_POOL_TOKEN

PGCR gate, GETNEXT_CONTAINER function

The GETNEXT_CONTAINER function of the PGCR gate is used to get the next container in a browse of containers.

Input parameters

BROWSE_TOKEN
is a browse token referencing the next container in a browse of containers.

Output parameters

[CCSID]
is the coded character set identifier of the character data in the container.

[CONTAINER_NAME]
is the name of the container.

[CONTAINER_TOKEN]
is a token referencing the container.

[DATA_LENGTH]
is the length, in bytes, of the data in the container.

[DATATYPE]
is the format of the data in the container. DATATYPE can have either of these values:

CHAR|BIT

[GENERATION_NUMBER]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

[INITIAL_GENERATION]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

[TYPE]
whether the container is visible only to CICS, or to user programs as well. TYPE can have either of these values:

CICS|USER

[USERACCESS]
whether the container can be updated by API commands. USERACCESS can have either of these values:

READONLY|ANY

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END INVALID_BROWSE_TOKEN

PGCR gate, INQUIRE_BROWSE_CONTEXT function

The INQUIRE_BROWSE__CONTEXT function of the PGCR gate is used to

Input parameters

BROWSE_TOKEN
is a browse token referencing the next container in a browse of containers.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN

PGCR gate, INQUIRE_CONTAINER function

The INQUIRE_CONTAINER function of the PGCR gate is used to retrieve the attributes of a container.

CCSID is the coded character set identifier that the character data in the container is stored in.

Input parameters

[CALLER]
is the call part of an API call. It can have either of these values:
EXEC|SYSTEM

CONTAINER_NAME
is the name of the container to be enquired upon.

POOL_TOKEN
is a token referencing the container pool to which the container belongs.

[TYPE]
whether the container is visible only to CICS, or to user programs as well. TYPE can have either of these values:
CICS|USER

Output parameters

[CCSID]
is the coded character set identifier of the character data in the container.

[CONTAINER_TOKEN]
is a token referencing the container.

[DATATYPE]
is the format of the data. It can have either of these values:

CHAR|BIT

[DATA_LENGTH]
is the length, in bytes, of the data in the container.

[GENERATION_NUMBER]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

[INITIAL_GENERATION]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

[USERACCESS]
whether USER containers can be updated by API commands. USERACCESS can have either of these values:

READONLY|ANY

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CONTAINER_NOT_FOUND
INVALID	INVALID_CONTAINER_TOKEN INVALID_PARAMETERS INVALID_POOL_TOKEN

PGCR gate, INQUIRE_CONTAINER_BY_TOKEN function

The INQIRE_CONTAINER_BY_TOKEN function of the PGCR gate is used to retrieve the attributes of a container by means of a token.

Input parameters

CONTAINER_TOKEN
is a token referencing the container to be enquired upon.

Output parameters

[CCSID]
is the coded character set identifier of the character data in the container.

[CONTAINER_NAME]
is the name of the container.

[DATA_LENGTH]
is the length, in bytes, of the data in the container.

[DATATYPE]
is the format of the data. It can have either of these values:

CHAR|BIT

[GENERATION_NUMBER]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

[INITIAL_GENERATION]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

[TYPE]
whether the container is visible only to CICS, or to user programs as well. TYPE can have either of these values:

`CICS|USER`

[USERACCESS]
whether USER containers can be updated by API commands. USERACCESS can have either of these values:

`READONLY|ANY`

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

REASON
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CONTAINER_NOT_FOUND
INVALID	INVALID_CONTAINER_TOKEN INVALID_PARAMETERS

PGCR gate, MOVE_CONTAINER function

The MOVE_CONTAINER function of the PGCR gate is used to move a container from one container pool to another. Both pools must already have been created. If the TO_POOL_TOKEN is not specified, the container is not moved to a different pool but is renamed to the value of AS_CONTAINER_NAME.

Input parameters

[AS_CONTAINER_NAME]
is the name by which the container is to be known in the target container pool.

[CALLER]
is the call part of an API call. It can have either of these values:

`EXEC|SYSTEM`

[CONTAINER_NAME]
is the name of the container to be moved.

[CONTAINER_TOKEN]
is a token representing the container to be moved.

[POOL_TOKEN]
is a token referencing the container pool from which the container is to be moved.

[TO_POOL_TOKEN]
is a token referencing the container pool to which the container is to be moved.

[TYPE]
whether the container is visible only to CICS, or to user programs as well. TYPE can have either of these values:

`CICS|USER`

Output parameters

[CONTAINER_TOKEN_OUT]
is a token representing the moved container in the target container pool.

[GENERATION_NUMBER]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

[INITIAL_GENERATION]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CONTAINER_NOT_FOUND INVALID_AS_CONTAINER_NAME READONLY_AS_CONTAINER READONLY_CONTAINER
INVALID	INVALID_CONTAINER_TOKEN INVALID_PARAMETERS INVALID_POOL_TOKEN INVALID_TO_POOL_TOKEN

PGCR gate, PUT_CONTAINER function

The PUT_CONTAINER function of the PGCR gate is used to put data into a container from an area provided by the caller.

Input parameters

[CALLER]
is the call part of an API call. It can have either of these values:

EXEC|SYSTEM

[CCSID]
is the coded character set identifier (CCSID) of the character data to be put into the container.

[CONTAINER_NAME]
is the name of the container.

[CONTAINER_TOKEN]
is a token referencing an existing container into which the data is to be put.

[CONVERT]
whether character data in the container should be converted. It can have either of these values:

YES|NO

The default value is YES.

[DATATYPE]
is the format of the data. It can have either of these values:

CHAR|BIT

ITEM_DATA
The address and length of the put data.

[POOL_TOKEN]
is a token referencing the container pool to which the container belongs.

[PUT_TYPE]
whether the PUT data should be appended to the current contents of the container or replace the current contents. PUT_TYPE can have either of these values:

`APPEND|REPLACE`

[TYPE]
whether the container is visible only to CICS, or to user programs as well. TYPE can have either of these values:

`CICS|USER`

[USERACCESS]
whether USER containers can be updated by API commands. USERACCESS can have either of these values:

`READONLY|ANY`

Output parameters

[CONTAINER_TOKEN_OUT]
is a token referencing the container.

[GENERATION_NUMBER]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

[INITIAL_GENERATION]
Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CCSID_INVALID DATATYPE_CHANGE INVALID_CONTAINER_NAME LENGTH_ERROR, READONLY_CONTAINER
INVALID	INVALID_CONTAINER_TOKEN INVALID_PARAMETERS INVALID_POOL_TOKEN

PGCR gate, SET_CONTAINER function

The SET_CONTAINER function of the PGCR gate is used to change the attributes of a container.

Input parameters

[CONTAINER_NAME]
is the name of the container.

[CONTAINER_TOKEN]
is a token referencing the container whose attributes are to be changed.

[POOL_TOKEN]
is a token referencing the container pool to which the container belongs.

[TYPE]
whether the container is visible only to CICS, or to user programs as well. TYPE can have either of these values:

CICS|USER

[USERACCESS]
whether USER containers can be updated by API commands. USERACCESS can have either of these values:

READONLY|ANY

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CONTAINER_NOT_FOUND
INVALID	INVALID_CONTAINER_TOKEN INVALID_PARAMETERS INVALID_POOL_TOKEN

PGCR gate, STARTBR_CONTAINER function

The STARTBR_CONTAINER function of the PGCR gate is used to initiate a browse of the containers in a specified container pool.

Input parameters

POOL_TOKEN
is a token referencing the container pool to be browsed.

[CALLER]
is the call part of an API call. It can have either of these values:

EXEC|SYSTEM

Output parameters

BROWSE_TOKEN
is a browse token referencing a container in the container pool. This container is the first in the browse list.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_POOL_TOKEN

PGCR gate, TRACE_CONTAINERS function

The TRACE_CONTAINER function of the PGCR gate is used to initiate a trace of the containers in a specified channel.

Input parameters

CHANNEL_TOKEN
is a token referencing the channel whose containers are to be traced.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_CHANNEL_TOKEN

PGDD gate, DEFINE_PROGRAM function

The DEFINE_PROGRAM function of the PGDD gate is used to define a program resource.

Input parameters

Note: Specify either the PROGRAM_NAME parameter or the CATALOG_ADDRESS parameter, not both.

PROGRAM_NAME
is the name of the program resource to be defined.

CATALOG_ADDRESS
is the token identifying the program resource to be defined.

[CEDF_STATUS]
indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF). It can have either of these values:

CEDF|NOCEDF

[LANGUAGE_DEFINED]
is the language to be defined for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|LE370|PLI|NOT_DEFINED

[AVAIL_STATUS]
defines whether (ENABLED) or not (DISABLED) the program can be used. It can have either of these values:

ENABLED|DISABLED

[MODULE_TYPE]
is the type of program resource to be defined: It can have any of these values:

PROGRAM|MAPSET|PARTITIONSET

[DATA_LOCATION]
defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). The DATALOCATION options are independent from the addressing mode of the link-edited program. It can have either of these values:

ANY|BELOW

[EXECUTION_SET]
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:

FULLAPI|DPLSUBSET

[REMOTE_PROGID]
is the name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value).

[REMOTE_SYSID]
is the name of a remote CICS region if you want CICS to ship a distributed program link (DPL) request to another CICS region.

[REMOTE_TRANID]
is the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.

[EXECUTION_KEY]
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

CICS|USER

Note: If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into extended the read-only DSA(ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

[PROGRAM_TYPE]
is the type of program. It can have any of these values:

PRIVATE|SHARED|TYPE_ANY

[PROGRAM_USAGE]
defines whether the program is to be used as a CICS nucleus program or as a user application program. It can have either of these values:

NUCLEUS|APPLICATION

[PROGRAM_ATTRIBUTE]
defines the residence status of the program, and when the storage for this program is released. It can have any of these values:

RESIDENT|REUSABLE|TRANSIENT|RELOAD|TEST

[REQUIRED_AMODE]
is the addressing mode of the program. It can have any of these values:

24|31|AMODE_ANY

[REQUIRED_RMODE]
is the residence mode of the program. It can have any of these values:

24|RMODE_ANY

[DYNAMIC_STATUS]
indicates whether or not a request to LINK to the program may be dynamically routed. It can have either of these values:

DYNAMIC|NOTDYNAMIC

[CONCURRENCY]
indicates whether the program is threadsafe or only quasi-reentrant. It can have either of these two values:

THREADSAFE|QUASIRENT

[JVM]
indicates whether or not the program is to be executed under the control of a JVM (Java Virtual Machine). It can have either of these values:

YES|NO

[JVM_CLASS]
is the name of the main class in a Java program to be run under the control of a JVM.

[HOTPOOL]
indicates whether or not the Java program object is to be run in a preinitialized Language Environment enclave reused by multiple invocations of the program, under control of an H8 TCB. It can have either of these two values:

YES|NO

[JVM_PROFILE]
specifies the name of the data set member that contains the JVM profile.. The named profile provides the attributes of the JVM that is needed to execute the program.

[MULTITCB]
is reserved for future use

[OPENAPI]
is reserved for future use

[NEW_PROGRAM_TOKEN]
is the Loader Domain token for the program

Output parameters

NEW_PROGRAM_TOKEN
is the token assigned to program.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND CATALOG_NOT_OPERATIONAL CATALOG_ERROR INSUFFICIENT_STORAGE LOCK_ERROR
EXCEPTION	PROGRAM_ALREADY_DEFINED PROGRAM_IN_USE
INVALID	INVALID_CATALOG_ADDRESS INVALID_FUNCTION INVALID_MODE_COMBINATION INVALID_PROGRAM_NAME INVALID_TYPE_ATTRIB_COMBIN

PGDD gate, DELETE_PROGRAM function

The DELETE_PROGRAM function of the PGDD gate is used to delete a program resource.

Input parameters

PROGRAM_NAME
is the name of the program resource to be defined.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOCK_ERROR
EXCEPTION	PROGRAM_NOT_DEFINED PROGRAM_NAME_STARTS_DFH PROGRAM_IS_URM PROGRAM_IN_USE
INVALID	INVALID_FUNCTION

PGEX gate, INITIALIZE_EXIT function

The INITIALIZE_EXIT function of the PGEX gate is used to initialize an exit program.

Input parameters

PROGRAM_NAME
is the name, 1 through 8 alphanumeric characters, of the program to be initialized.

LOAD_PROGRAM
defines whether or not the program is to be loaded when initialized. It can have either of these values:

YES|NO

SYSTEM_AUTOINSTALL
defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition. It can have either of these values:

YES|NO

[LPA_ELIGIBLE]
defines whether or not the program can be loaded into the MVS link pack area (LPA). It can have either of these values:

YES|NO

Output parameters

PROGRAM_TOKEN
is the token assigned to program.

[ENTRY_POINT]
is the token defining the entry point of the program.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	AUTOINSTALL_URM_FAILED AUTOINSTALL_MODEL_NOT_DEF AUTOINSTALL_INVALID_DATA AUTOINSTALL_FAILED JVM_PROGRAM PROGRAM_NOT_AUTHORIZED PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE REMOTE_PROGRAM
INVALID	INVALID_INITIALIZE_REQUEST INVALID_FUNCTION

PGEX gate, TERMINATE_EXIT function

The TERMINATE_EXIT function of the PGEX gate is used to terminate an exit program.

Input parameters

PROGRAM_TOKEN
is the token identifying the program to be terminated.

RELEASE_PROGRAM
defines whether or not the program is to be released when terminated. It can have either of these values:

YES|NO

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

RESPONSE	Possible REASON values
EXCEPTION	PROGRAM_NOT_AUTHORIZED PROGRAM_NOT_DEFINED PROGRAM_NOT_IN_USE PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE
INVALID	INVALID_PROGRAM_TOKEN INVALID_FUNCTION

PGHM gate, SET_CONDITIONS function

The SET_CONDITIONS function of the PGHM gate is used to process for user EXEC CICS HANDLE CONDITION commands, and to save the details of the condition into the current condition handle table.

Input parameters

IDENTIFIERS
is the token identifying the conditions to be handled.

LABELS_FLAGS
is the token identifying the number of conditions in this command that have associated labels.

[LABELS]
is the token identifying the condition labels (the locations within the program to be branched to if the condition occurs).

[LANGUAGE]
is the program language. It can have any of these values:

ASSEMBLER|C370|COBOL|LE370|PLI

[CURRENT_EXECUTION_KEY]
is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE CONDITION command was issued).

[USERS_RSA_POINTER]
is the address of the user program Register Save Area into which the program's registers are saved at each EXEC CICS command execution.

[AMODE]
is the addressing mode (24–bit or 31–bit) of the program at the time the handle command was driven.

Output parameters

[FASTPATH_FLAGS]
identifies the fastpath flag settings for the following conditions handled by the user: RDATT, WRBRK, EOF, NOSPACE, QBUSY, NOSTG, ENQBUSY, NOJBUFSP, SIGNAL, OVERFLOW, SYSBUSY, SESSBUSY.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION MISSING_PARAMETER

PGHM gate, INQ_CONDITION function

The INQ_CONDITION function of the PGHM gate is invoked when a condition has occurred, and returns to the caller about details of the condition for user EXEC CICS HANDLE CONDITION commands.

Input parameters

CONDITION
is an 8-bit value identifying the condition.

Output parameters

STATUS
identifies the status of the condition. It can have any of these values:

DEFAULT|HANDLED|IGNORED

[LABEL] is the token identifying the condition label within the program to be branched to if the condition occurs.

[LANGUAGE] is the program language. It can have any of these values:

ASSEMBLER|C370|COBOL|LE370|PLI

[CURRENT_EXECUTION_KEY]
is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE CONDITION command was issued).

[USERS_RSA_POINTER]
is the address of the user program Register Save Area into which the program's registers are saved at each EXEC CICS command execution.

[PROGRAM_MASK]
identifies the program mask at the time the HANDLE CONDITION command was executed.

[GOTOL] is the token identifying the condition label within the program to be branched to if the condition is ignored.

[ABEND_CODE] is the four-character abend code to be issued if CICS drives the system default, which is to abend the transaction.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION MISSING_PARAMETER

PGHM gate, IGNORE_CONDITIONS function

The IGNORE_CONDITIONS function of the PGHM gate is used to ignore the conditions for user EXEC CICS IGNORE CONDITION commands.

Input parameters

IDENTIFIERS is the token identifying the conditions to be ignored.

Output parameters

[FASTPATH_FLAGS]
identifies the fastpath flag settings for the conditions.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION MISSING_PARAMETER

PGHM gate, SET_AIDS function

The SET_AIDS function of the PGHM gate is invoked in response to a user EXEC CICS HANDLE AID command, and saves the details of the handle into the current aid Handle Table.

Input parameters

IDENTIFIERS is the token identifying the aids to be handled.

LABELS_FLAGS is the token identifying the number of aids in this command that have associated labels.

[LABELS] is the token identifying the condition labels (the locations within the program to be branched to if the aid occurs).

[LANGUAGE] is the program language. It can have any of these values:

ASSEMBLER|C370|COBOL|LE370|PLI

[CURRENT_EXECUTION_KEY]
is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE AID command was issued).

[USERS_RSA_POINTER]
is the address of the user program Register Save Area into which the program's registers are saved at each EXEC CICS command execution.

[AMODE] is the addressing mode (24–bit or 31–bit) of the program at the time the handle command was driven.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION MISSING_PARAMETER

PGHM gate, INQ_AID function

The INQ_AID function of the PGHM gate is invoked when an aid has occurred, and returns to the caller details of the handle aid for user EXEC CICS HANDLE AID commands.

Input parameters

AID is an 8-bit value identifying the aid.

Output parameters

[LABEL] is the token identifying the condition label within the program to be branched to if the aid occurs.

[LANGUAGE] is the program language. It can have any of these values:

ASSEMBLER|C370|COBOL|LE370|PLI

[CURRENT_EXECUTION_KEY]
is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE AID command was issued).

[USERS_RSA_POINTER]
is the address of the user program Register Save Area into which the program's registers are saved at each EXEC CICS command execution.

[PROGRAM_MASK]
identifies the program mask at the time the HANDLE CONDITION command was executed.

[GOTOL] is the token identifying the condition label within the program to be branched to if the condition is ignored.

[STATUS] identifies the status of the AID. It can have either of these values:

SYSTEM_DEFAULT|HANDLED

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION MISSING_PARAMETER

PGHM gate, SET_ABEND function

The SET_ABEND function of the PGHM gate is invoked in response to a user EXEC CICS HANDLE ABEND command, and saves the details of the handle into the current abend Handle Table.

Input parameters

OPERATION
identifies what is to be done if the abend occurs. It can have any of these values:

HANDLE|CANCEL|RESET

Note: Specify either the LABEL parameter or the PROGRAM parameter, not both.

[LABEL]
is the token identifying the condition label within the program to be branched to if the abend occurs.

[PROGRAM]
is the name of the program to which control will be passed if the abend occurs.

[LANGUAGE]
is the program language. It can have any of these values:

ASSEMBLER|C370|COBOL|LE370|PLI

[CURRENT_EXECUTION_KEY]
is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE ABEND command was issued).

[USERS_RSA_POINTER]
is the address of the user program Register Save Area into which the program's registers are saved at each EXEC CICS command execution.

[AMODE]
is the addressing mode (24–bit or 31–bit) of the program at the time the handle command was driven.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION MISSING_PARAMETER

PGHM gate, INQ_ABEND function

The INQ_ABEND function of the PGHM gate is invoked when an abend has occurred, and returns to the caller details of the handle abend for user EXEC CICS HANDLE AID commands.

Input parameters

None.

Output parameters

STATUS
identifies the status of the condition. It can have either of these values:

SYSTEM_DEFAULT|HANDLED

[LABEL]
is the token identifying the condition label within the program branched to when the abend occurred.

[PROGRAM]
is the name of the program to which control was passed when the abend occurred.

[LANGUAGE]
is the program language. It can have any of these values:

ASSEMBLER|C370|COBOL|LE370|PLI

[CURRENT_EXECUTION_KEY]
is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE AID command was issued).

[USERS_RSA_POINTER]
is the address of the user program Register Save Area into which the program's registers are saved at each EXEC CICS command execution.

[PROGRAM_MASK]
identifies the program mask at the time the HANDLE CONDITION command was executed.

[GOTOL]
is the token identifying the condition label within the program to be branched to if the condition is ignored.

[HANDLE_COUNT]
is the number of times that this abend code has been handled.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION MISSING_PARAMETER

PGHM gate, PUSH_HANDLE function

The PUSH_HANDLE function of the PGHM gate is invoked for a user EXEC CICS PUSH command.

Input parameters

None.

Output parameters

[FASTPATH_FLAGS]
identifies the fastpath flag settings for the conditions.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION MISSING_PARAMETER

PGHM gate, POP_HANDLE function

The POP_HANDLE function of the PGHM gate is invoked for a user EXEC CICS POP command.

Input parameters

None.

Output parameters

[FASTPATH_FLAGS]
identifies the fastpath flag settings for the conditions.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	NO_PREVIOUS_PUSH
INVALID	INVALID_FUNCTION MISSING_PARAMETER

PGHM gate, FREE_HANDLE_TABLES function

The FREE_HANDLE_TABLES function of the PGHM gate is invoked by CICS during program termination processing and frees all storage relating to the Handle State for that program level.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION MISSING_PARAMETER

PGHM gate, CLEAR_LABELS function

The CLEAR_LABELS function of the PGHM gate is invoked by CICS during XCTL processing and frees all storage relating to the Handle State for that program (except for the initial default state) and removes all user-defined label handles.

Input parameters

None.

Output parameters

[FASTPATH_FLAGS]
identifies the fastpath flag settings for the conditions.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION MISSING_PARAMETER

PGIS gate, INQUIRE_PROGRAM function

The INQUIRE_PROGRAM function of the PGIS gate is used to inquire about attributes of a program.

Input parameters

Note: Specify either the PROGRAM_NAME parameter or the PROGRAM_TOKEN parameter, not both.

PROGRAM_NAME
is the name of the program.

PROGRAM_TOKEN
is the token identifying the program.

Output parameters

[CEDF_STATUS]
indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF) It can have any of these values:

CEDF|NOCEDF|NOT_APPLIC

[HOLD_STATUS]
is the hold status of the program (that is, for how long the program is to be loaded). It can have any of these values:

TASK_LIFE|CICS_LIFE|NOT_APPLIC

[LOAD_STATUS]
is the load status of the program (that is, whether or not the program can be loaded). It can have any of these values:

LOADABLE|NOT_LOADABLE|NOT_LOADED|NOT_APPLIC

[INSTALL_TYPE]
is the method used to install the PROGRAM resource definition. It can have any of these values:

RDO|CATALOG|GROUPLIST|AUTO|SYSAUTO|MANUAL

[LANGUAGE_DEFINED]
is the language defined for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|LE370|PLI|
NOT_DEFINED|NOT_APPLIC

[LANGUAGE_DEDUCED]
is the language deduced by CICS for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|COBOL2|JAVA|LE370|PLI|
NOT_DEDUCED|NOT_APPLIC

[AVAIL_STATUS]
defines whether (ENABLED) or not (DISABLED) the program can be used. It can have either of these values:

ENABLED|DISABLED

[MODULE_TYPE]
is the type of program resource to be defined: It can have any of these values:

PROGRAM|MAPSET|PARTITIONSET

[DATA_LOCATION]
defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). The DATALOCATION options are independent from the addressing mode of the link-edited program. It can have either of these values:

ANY|BELOW|NOT_APPLIC

[EXECUTION_SET]
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:

FULLAPI|DPLSUBSET|NOT_APPLIC

[REMOTE_PROGID]
is the name by which the program is known in the remote CICS region. If you specify

REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value.

[REMOTE_SYSID]
is the name of a remote CICS region if you want CICS to ship a distributed program link (DPL) request to another CICS region.

[REMOTE_TRANID]
is the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.

[EXECUTION_KEY]
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

`CICS|USER|NOT_APPLIC`

Note: If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into extended the read-only DSA(ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

[PROGRAM_TYPE]
is the type of program. It can have any of these values:

`PRIVATE|SHARED|TYPE_ANY|NOT_APPLIC`

[PROGRAM_USAGE]
defines whether the program is to be used as a CICS nucleus program or as a user application program. It can have either of these values:

`NUCLEUS|APPLICATION`

[PROGRAM_ATTRIBUTE]
defines the residence status of the program, and when the storage for this program is released. It can have any of these values:

`RESIDENT|REUSABLE|TRANSIENT|RELOAD|TEST`

[SPECIFIED_AMODE]
is the addressing mode of the program. It can have any of these values:

`24|31|AMODE_ANY|AMODE_NOT_SPECIFIED`

[SPECIFIED_RMODE]
is the residence mode of the program. It can have any of these values:

`24|RMODE_ANY|RMODE_NOT_SPECIFIED`

[PROGRAM_LENGTH]
is the length of the program. returned by the loader domain on the ACQUIRE_PROGRAM call.

[PROGRAM_USE_COUNT]
is the number of times that the program has been used.

[PROGRAM_USER_COUNT]
is the number of different users that have invoked the program.

[LOAD_POINT]
is the load point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

[ENTRY_POINT]
is the entry point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

[LOCATION]
defines where the program resides. It can have any of these values:

`CDSA|ECDSA|SDSA|ESDSA|RDSA|ERDSA|LPA|ELPA|NONE`

[ACCESS]
is the type of access for the program. It can have any of these values:

`USER|CICS|READ_ONLY|NONE`

[REMOTE_DEFINITION]
indicates whether the program is defined as remote or local. It can take the values:

`REMOTE|LOCAL`

[NEW_PROGRAM_TOKEN]
is the loader domain token for the program

[DYNAMIC_STATUS]
indicates whether or not a request to LINK to the program may be dynamically routed. It can have either of these values:

DYNAMIC|NOTDYNAMIC

[CONCURRENCY]
indicates whether the program is threadsafe or only quasi-reentrant. It can have either of these two values:

THREADSAFE|QUASIRENT

[JVM]
indicates whether or not the program is to be executed under the control of a JVM (Java Virtual Machine). It can have either of these values:

YES|NO

[JVM_CLASS]
is the name of the main class in a Java program to be run under the control of a JVM.

[HOTPOOL]
indicates whether or not the Java program object is to be run in a preinitialized Language Environment enclave reused by multiple invocations of the program, under control of an H8 TCB. It can have either of these two values:

YES|NO

[JVM_PROFILE]
specifies the name of the JVM profile. The named profile provides the attributes of the JVM that is needed to execute the program.

[JVMPROGRAM_USE_COUNT]
For Java programs to be run under the control of a JVM, the number of times the program has been used.

[RUNTIME_ENVIRONMENT]
indicates the runtime environment used for the execution of this program. It can take the values

JVM_RUNTIME|LE370_RUNTIME|NON_LE370_RUNTIME|
UNKNOWN_RUNTIME|NOT_APPLIC

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOCK_ERROR
EXCEPTION	PROGRAM_NOT_DEFINED_TO_LD PROGRAM_NOT_DEFINED_TO_PG
INVALID	INVALID_PROGRAM_TOKEN

PGIS gate, INQUIRE_CURRENT_PROGRAM function

The INQUIRE_CURRENT_PROGRAM function of the PGIS gate is used to inquire about the current attributes of a program (for the current invocation of the program).

Input parameters

None.

Output parameters

[CEDF_STATUS]
indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF) It can have any of these values:

CEDF|NOCEDF|NOT_APPLIC

[HOLD_STATUS]
is the hold status of the program (that is, for how long the program is to be loaded). It can have any of these values:

`TASK_LIFE|CICS_LIFE|NOT_APPLIC`

[LOAD_STATUS]
is the load status of the program (that is, whether or not the program can be loaded). It can have any of these values:

`LOADABLE|NOT_LOADABLE|NOT_LOADED|NOT_APPLIC`

[INSTALL_TYPE]
is the method used to install the PROGRAM resource definition. It can have any of these values:

`RDO|CATALOG|GROUPLIST|AUTO|SYSAUTO|MANUAL`

[LANGUAGE_DEFINED]
is the language defined for the program. It can have any of these values:

`ASSEMBLER|C370|COBOL|LE370|PLI|`
`NOT_DEFINED|NOT_APPLIC`

[LANGUAGE_DEDUCED]
is the language deduced by CICS for the program. It can have any of these values:

`ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI|`
`NOT_DEDUCED|NOT_APPLIC`

[AVAIL_STATUS]
defines whether (ENABLED) or not (DISABLED) the program can be used. It can have either of these values:

`ENABLED|DISABLED`

[MODULE_TYPE]
is the type of program resource to be defined: It can have any of these values:

`PROGRAM|MAPSET|PARTITIONSET`

[DATA_LOCATION]
defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). The DATALOCATION options are independent from the addressing mode of the link-edited program. It can have either of these values:

`ANY|BELOW|NOT_APPLIC`

[EXECUTION_SET]
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have any of these values:

`FULLAPI|DPLSUBSET|NOT_APPLIC`

[REMOTE_DEFINITION]
defines whether the program is local or remote. It can have either of these values:

`LOCAL|REMOTE`

[REMOTE_PROGID]
is the name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value).

[REMOTE_SYSID]
is the name of a remote CICS region if you want CICS to ship a distributed program link (DPL) request to another CICS region.

[REMOTE_TRANID]
is the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.

[EXECUTION_KEY]
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have any of these values:

CICS|USER|NOT_APPLIC

Note: If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into extended the read-only DSA(ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

NEW_PROGRAM_TOKEN
is the token assigned to program.

[CURRENT_PROGRAM_NAME]
is the current name of the program.

[INVOKING_PROGRAM_NAME]
is the name of the program invoking this program.

[RETURN_PROGRAM_NAME]
is the name of the program to which control will be returned when this program has ended.

[CURRENT_CEDF_STATUS]
indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF) It can have either of these values:

CEDF|NOCEDF

[CURRENT_EXECUTION_SET]
indicates whether the program is running with or without the API restrictions of a DPL program. It can have any of these values:

FULLAPI|DPLSUBSET

[CURRENT_ENVIRONMENT]
indicates the current environment in which the program is running. It can have any of these values:

EXEC|GLUE|PLT|SYSTEM|TRUE|URM

[CURRENT_AMODE]
is the addressing mode of the program. It can have either of these values:

24|31

[CURRENT_LOAD_POINT]
is the current load point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

[CURRENT_ENTRY_POINT]
is the current entry point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

[CURRENT_PROGRAM_LENGTH]
is the length of the current program in bytes, as returned by the Loader Domain on the AQUIRE_PROGRAM call.

[INVOKING_ENVIRONMENT]
is the environment in which the program invoking this program was executing. It can have any of these values:

EXEC|GLUE|PLT|SYSTEM|TRUE|URM

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOCK_ERROR
EXCEPTION	NO_CURRENT_PROGRAM

PGIS gate, SET_PROGRAM function

The SET_PROGRAM function of the PGIS gate is used to set the characteristics of a program when it is loaded.

Input parameters

note.Specify either the PROGRAM_NAME parameter or the PROGRAM_TOKEN parameter, not both.

PROGRAM_NAME is the name of the program.

PROGRAM_TOKEN
is the token identifying the program.

[CEDF_STATUS]
indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF) It can have either of these values:

CEDF|NOCEDF

[AVAIL_STATUS]
defines whether (ENABLED) or not (DISABLED) the program can be used. It can have either of these values:

ENABLED|DISABLED

[EXECUTION_SET]
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:

FULLAPI|DPLSUBSET

[EXECUTION_KEY]
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

CICS|USER

Note: If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into extended the read-only DSA(ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

[PROGRAM_TYPE]
is the type of program. It can have any of these values:

PRIVATE|SHARED|TYPE_ANY

[PROGRAM_USAGE]
defines whether the program is to be used as a CICS nucleus program or as a user application program. It can have either of these values:

NUCLEUS|APPLICATION

[PROGRAM_ATTRIBUTE]
defines the residence status of the program, and when the storage for this program is released. It can have any of these values:

RESIDENT|REUSABLE|TRANSIENT|RELOAD|TEST

[REQUIRED_AMODE]
is the addressing mode of the program. It can have any of these values:

24|31|AMODE_ANY

[REQUIRED_RMODE]
is the residence mode of the program. It can have any of these values:

24|RMODE_ANY

[JVM] indicates whether or not the program is to be executed under the control of a JVM (Java Virtual Machine). It can have either of these values:

YES|NO

[JVM_CLASS] is the name of the main class in a Java program to be run under the control of a JVM.

[JVM_PROFILE]
is the name of the JVM profile that provides the attributes of the JVM which is needed to execute the program.

[HOTPOOL]
indicates whether or not the Java program object is to be run in a preinitialized Language Environment enclave reused by multiple invocations of the program, under control of an H8 TCB. It can have either of these two values:

YES|NO

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND CATALOG_NOT_OPERATIONAL CATALOG_ERROR LOCK_ERROR INSUFFICIENT_STORAGE
EXCEPTION	PROGRAM_NOT_DEFINED_TO_PG CEDF_STATUS_NOT_FOR_REMOTE CEDF_STATUS_NOT_FOR_MAPSET CEDF_STATUS_NOT_FOR_PTNSET EXEC_SET_NOT_FOR_REMOTE EXEC_SET_NOT_FOR_MAPSET EXEC_SET_NOT_FOR_PTNSET EXEC_KEY_NOT_FOR_REMOTE EXEC_KEY_NOT_FOR_MAPSET EXEC_KEY_NOT_FOR_PTNSET PROG_TYPE_NOT_FOR_REMOTE PROGRAM_NOT_DEFINED_TO_LD PROGRAM_NOT_FOUND JVM_BUT_NO_JVMCLASS HOTPOOL_NOT_FOR_REMOTE HOTPOOL_NOT_FOR_MAPSET HOTPOOL_NOT_FOR_PTNSET
INVALID	INVALID_MODE_COMBINATION INVALID_PROGRAM_NAME INVALID_PROGRAM_TOKEN INVALID_TYPE_ATTRIB_COMBIN

PGIS gate, START_BROWSE_PROGRAM function

The START_BROWSE_PROGRAM function of the PGIS gate is used to start browsing through program definitions, optionally starting at the given program definition.

Input parameters

[PROGRAM_NAME]
is the optional name of the program definition at which you want to start browsing.

[TASK_RELATED]
indicates whether or not the browse is task-related. If it is task-related, storage will be obtained from the CICS storage class rather than the directory browse subpool. The default is YES.

Output parameters

BROWSE_TOKEN
is a token identifying the program definition being browsed.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INVALID_DIRECTORY LOCK_ERROR

PGIS gate, GET_NEXT_PROGRAM function

The GET_NEXT_PROGRAM function of the PGIS gate is used to get the next program definition to be browse.

Input parameters

BROWSE_TOKEN is a token identifying the program definition to be browsed.

Output parameters

[CEDF_STATUS]
indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF) It can have any of these values:

CEDF|NOCEDF|NOT_APPLIC

[HOLD_STATUS]
is the hold status of the program (that is, for how long the program is to be loaded). It can have any of these values:

TASK_LIFE|CICS_LIFE|NOT_APPLIC

[LOAD_STATUS]
is the load status of the program (that is, whether or not the program can be loaded). It can have any of these values:

LOADABLE|NOT_LOADABLE|NOT_LOADED|NOT_APPLIC

[INSTALL_TYPE]
is the method used to install the PROGRAM resource definition. It can have any of these values:

RDO|CATALOG|GROUPLIST|AUTO|SYSAUTO|MANUAL

[LANGUAGE_DEFINED]
is the language defined for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|LE370|PLI|
NOT_DEFINED|NOT_APPLIC

[LANGUAGE_DEDUCED]
is the language deduced by CICS for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI|
NOT_DEDUCED|NOT_APPLIC

[AVAIL_STATUS]
defines whether (ENABLED) or not (DISABLED) the program can be used. It can have either of these values:

ENABLED|DISABLED

[MODULE_TYPE]
is the type of program resource to be defined: It can have any of these values:

PROGRAM|MAPSET|PARTITIONSET

[DATA_LOCATION]
defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). The DATALOCATION options are independent from the addressing mode of the link-edited program. It can have either of these values:

ANY|BELOW|NOT_APPLIC

[EXECUTION_SET]
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:

FULLAPI|DPLSUBSET|NOT_APPLIC

[REMOTE_PROGID]
is the name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value.

[REMOTE_SYSID]
is the name of a remote CICS region if you want CICS to ship a distributed program link (DPL) request to another CICS region.

[REMOTE_TRANID]
is the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.

[EXECUTION_KEY]
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

CICS|USER|NOT_APPLIC

Note: If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into extended the read-only DSA(ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

[PROGRAM_TYPE]
is the type of program. It can have any of these values:

PRIVATE|SHARED|TYPE_ANY|NOT_APPLIC

[PROGRAM_USAGE]
defines whether the program is to be used as a CICS nucleus program or as a user application program. It can have either of these values:

NUCLEUS|APPLICATION

[PROGRAM_ATTRIBUTE]
defines the residence status of the program, and when the storage for this program is released. It can have any of these values:

RESIDENT|REUSABLE|TRANSIENT|RELOAD|TEST

[SPECIFIED_AMODE]
is the addressing mode of the program. It can have any of these values:

24|31|AMODE_ANY|AMODE_NOT_SPECIFIED

[SPECIFIED_RMODE]
is the residence mode of the program. It can have any of these values:

24|RMODE_ANY|RMODE_NOT_SPECIFIED

[PROGRAM_LENGTH]
is the length of the program. returned by the loader domain on the ACQUIRE_PROGRAM call.

[PROGRAM_USE_COUNT]
is the number of times that the program has been used.

[PROGRAM_USER_COUNT]
is the number of different users that have invoked the program.

[LOAD_POINT] is the load point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

[ENTRY_POINT]
is the entry point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

[LOCATION] defines where the program resides. It can have any of these values:

CDSA|ECDSA|SDSA|ESDSA|RDSA|ERDSA|LPA|ELPA|NONE

[ACCESS is the type of access for the program. It can have any of these values:

USER|CICS|READ_ONLY|NONE

[REMOTE_DEFINITION]
indicates whether the program is defined as remote or local. It can take the values:

REMOTE|LOCAL

[NEW_PROGRAM_TOKEN]
is the loader domain token for the program

[DYNAMIC_STATUS]
indicates whether or not a request to LINK to the program may be dynamically routed. It can have either of these values:

DYNAMIC|NOTDYNAMIC

[CONCURRENCY]
indicates whether the program is threadsafe or only quasi-reentrant. It can have either of these two values:

THREADSAFE|QUASIRENT

[JVM] indicates whether or not the program is to be executed under the control of a JVM (Java Virtual Machine). It can have either of these values:

YES|NO

[JVM_CLASS] is the name of the main class in a Java program to be run under the control of a JVM.

[HOTPOOL] indicates whether or not the Java program object is to be run in a preinitialized Language Environment enclave reused by multiple invocations of the program, under control of an H8 TCB. It can have either of these two values:

YES|NO

[JVM_PROFILE]
specifies the name of the data set member that contains the JVM profile.. The named profile provides the attributes of the JVM that is needed to execute the program.

[RUNTIME_ENVIRONMENT]
indicates the runtime environment used for the execution of this program. It can take the values

JVM_RUNTIME|LE370_RUNTIME|NON_LE370_RUNTIME|
UNKNOWN_RUNTIME|NOT_APPLIC

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOCK_ERROR
EXCEPTION	INVALID_BROWSE_TOKEN END_LIST PROGRAM_NOT_DEFINED_TO_LD

PGIS gate, END_BROWSE_PROGRAM function

The END_BROWSE_PROGRAM function of the PGIS gate is used to end browsing through program definitions.

Input parameters

BROWSE_TOKEN is a token identifying the last program definition that was browsed.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOCK_ERROR
EXCEPTION	INVALID_BROWSE_TOKEN END_LIST

PGIS gate, REFRESH_PROGRAM function

The REFRESH_PROGRAM function of the PGIS gate is used to inform the loader domain that a new copy of a named program is now available for use in the relocatable program library.

Input parameters

PROGRAM_NAME is the name of the program being refreshed.

COPY indicates whether a NEWCOPY or PHASEIN function is required.

Output parameters

VERSION is the version of the program after the REFRESH_PROGRAM function call. It can have either of these values:

NEW|OLD

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOCK_ERROR
EXCEPTION	PROGRAM_LOADED_CICS_LIFE PROGRAM_NOT_DEFINED_TO_LD PROGRAM_NOT_DEFINED_TO_PG PROGRAM_NOT_FOUND REMOTE_PROGRAM PROGRAM_IN_USE

PGLD gate, LOAD_EXEC function

The LOAD_EXEC function of the PGLD gate is used to load a program in response to an EXEC CICS LOAD command.

Input parameters

PROGRAM_NAME is the name of the program being refreshed.

HOLD_LIFETIME
determines for how long the program is to be loaded; that is, for the life-time of CICS (or until explicitly deleted) or for the lifetime of the task (unless explicitly deleted by the task). It can have either of these values:

CICS_LIFE|TASK_LIFE|CALLER_MANAGED

Output parameters

LOAD_POINT is the current load point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

ENTRY_POINT is the current entry point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

PROGRAM_LENGTH
is the length of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

[LANGUAGE_TOKEN]
is a token representing the AP domain language block for the program.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE REMOTE_PROGRAM NOT_AUTHORIZED AUTOINSTALL_URM_FAILED AUTOINSTALL_INVALID_DATA AUTOINSTALL_MODEL_NOT_DEF AUTOINSTALL_FAILED NOT_INITIALIZED JVM_PROGRAM
INVALID	INVALID_FUNCTION

PGLD gate, LOAD function

The LOAD function of the PGLD gate is used to load a program in response to a CICS internal load request.

Input parameters

PROGRAM_NAME
is the name of the program being refreshed.

HOLD_LIFETIME
determines for how long the program is to be loaded; that is, for the life-time of CICS (or until explicitly deleted) or for the lifetime of the task (unless explicitly deleted by the task). It can have either of these values:

CICS_LIFE|TASK_LIFE

MODULE_TYPE
is the type of program to be loaded: It can have any of these values:

PROGRAM|MAPSET|PARTITIONSET

SYSTEM_AUTOINSTALL
defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition. It can have either of these values:

YES|NO

[LPA_ELIGIBLE]
defines whether or not the program can be loaded into the MVS link pack area (LPA). It can have either of these values:

YES|NO

[SUSPEND]
This option is passed to the LDLD call, and thence to SMGF. It specifies the action in the event of a storage shortage.YES, the default value, means that the task will be suspended until storage is available. NO means that the task will be abended.

Output parameters

LOAD_POINT
is the current load point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

ENTRY_POINT
is the current entry point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

[PROGRAM_LENGTH]
is the length of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE REMOTE_PROGRAM AUTOINSTALL_URM_FAILED AUTOINSTALL_INVALID_DATA AUTOINSTALL_MODEL_NOT_DEF AUTOINSTALL_FAILED JVM_PROGRAM
INVALID	INVALID_FUNCTION

PGLD gate, RELEASE_EXEC function

The RELEASE_EXEC function of the PGLD gate is used to release a program in response to an EXEC CICS RELEASE command.

Input parameters

PROGRAM_NAME is the name of the program being released.

[ENTRY_POINT]
must be provided on RELEASE_EXEC by the caller for a program loaded with caller-managed lifetime.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_AUTHORIZED NOT_INITIALIZED PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_IN_USE PROGRAM_NOT_LOADED PROGRAM_RELOAD_YES RELEASE_ISSUING_PROGRAM REMOTE_PROGRAM JVM_PROGRAM
INVALID	INVALID_FUNCTION

PGLD gate, RELEASE function

The RELEASE function of the PGLD gate is used by CICS internal modules to release a program in response previously loaded by a PGLD LOAD request.

Input parameters

PROGRAM_NAME is the name of the program being released.

[ENTRY_POINT]
must be provided on RELEASE_EXEC by the caller for a program loaded with caller-managed lifetime.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_IN_USE PROGRAM_NOT_LOADED PROGRAM_RELOAD_YES REMOTE_PROGRAM JVM_PROGRAM
INVALID	INVALID_FUNCTION

PGLE gate, LINK_EXEC function

The LINK_EXEC function of the PGLE gate is used to link to a program in response to a user EXEC CICS LINK command.

Input parameters

PROGRAM_NAME is the name of the program to be linked.

[COMMAREA] is the optional communications area to be made available to the linked program.

[HANDLE_ABEND_PGM]
defines whether or not the program is to run as an abend handler program. It can have either of these values:

YES|NO

[INPUTMSG] is a data area to be supplied to the linked program on its first execution of an EXEC CICS RECEIVE command.

[SYNCONRETURN]
defines whether or not a syncpoint is to be taken on return from the linked program. It can have either of these values:

YES|NO

[SYSEIB_REQUEST]
Specifies whether the EXEC CICS LINK had the SYSEIB translator option specified.

[FORCE_LOCAL]
indicates whether the program must execute locally.

Output parameters

[REMOTE_PROGRAM_NAME]
is the name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value).

[REMOTE_SYSID]
is the name of a remote CICS region if you want CICS to ship a distributed program link (DPL) request to another CICS region.

[REMOTE_TRANID]
is the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.

[ABEND_CODE] is the four-character abend code to be issued if there is an exception response with reason TRANSACTION_ABEND.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	AUTOINSTALL_FAILED AUTOINSTALL_INVALID_DATA AUTOINSTALL_MODEL_NOT_DEF AUTOINSTALL_URM_FAILED DESTRUCTIVE_OVERLAP INVALID_COMMAREA_ADDR INVALID_COMMAREA_LEN INVALID_INPUTMSG_LEN INVALID_TERMINAL_TYPE NOT_INITIALIZED PROGRAM_NOT_AUTHORIZED PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE REMOTE_PROGRAM TRANSACTION_ABEND DYNAMIC_PGM SECOND_JVM_PROGRAM SECOND_H8_PROGRAM JVMPOOL_DISABLED JVM_PROFILE_NOT_FOUND JVM_PROFILE_NOT_VALID SYSTEM_PROPERTIES_NOT_FND USER_CLASS_NOT_FOUND

PGLK gate, LINK function

The LINK function of the PGLK gate is used by CICS internal modules to link to a program.

Input parameters

PROGRAM_NAME is the name of the program being linked.

SYSTEM_AUTOINSTALL
defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition. It can have either of these values:

YES|NO

[LPA_ELIGIBLE]
defines whether or not the program can be loaded into the MVS link pack area (LPA). It can have either of these values:

YES|NO

[PARMLIST_PTR]
is the address of a parameter list passed by the CICS program initiating the PGLK link to the new program.

Output parameters

[ABEND_CODE] is the four-character abend code to be issued if there is an exception response with reason TRANSACTION_ABEND.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE REMOTE_PROGRAM AUTOINSTALL_URM_FAILED AUTOINSTALL_MODEL_NOT_DEF AUTOINSTALL_INVALID_DATA AUTOINSTALL_FAILED TRANSACTION_ABEND

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION

PGLK gate, LINK_PLT function

The LINK_PLT function of the PGLK gate is used by CICS internal modules to link to a program in the program list table.

Input parameters

PROGRAM_NAME is the name of the program being linked.

SYSTEM_AUTOINSTALL
defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition. It can have either of these values:

YES|NO

[LPA_ELIGIBLE]
defines whether or not the program can be loaded into the MVS link pack area (LPA). It can have either of these values:

YES|NO

Output parameters

[ABEND_CODE] is the four-character abend code to be issued if there is an exception response with reason TRANSACTION_ABEND.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE REMOTE_PROGRAM AUTOINSTALL_URM_FAILED AUTOINSTALL_MODEL_NOT_DEF AUTOINSTALL_INVALID_DATA AUTOINSTALL_FAILED TRANSACTION_ABEND
INVALID	INVALID_FUNCTION

PGLU gate, LINK_URM function

The LINK_URM function of the PGLU gate is used by CICS internal modules to link to a user-replaceable program.

Input parameters

PROGRAM_NAME is the name of the program to be linked.

SYSTEM_AUTOINSTALL
defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition. It can have either of these values:

YES|NO

[LPA_ELIGIBLE]
defines whether or not the program can be loaded into the MVS link pack area (LPA). It can have either of these values:

YES|NO

[COMMAREA] is the optional communications area to be made available to the linked program.

[CALLER_THREADSAFE]
indicates that the caller of the user-replaceable program is threadsafe, and so execution can continue on any TCB on return from the program: there is no need for PGLU to issue change_mode.

Output parameters

[ABEND_CODE] is the four-character abend code to be issued if there is an exception response with reason URM_ABEND.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE REMOTE_PROGRAM AUTOINSTALL_URM_FAILED AUTOINSTALL_MODEL_NOT_DEF AUTOINSTALL_INVALID_DATA AUTOINSTALL_FAILED INVALID_COMMAREA_LEN INVALID_COMMAREA_ADDR AMODE_ERROR URM_ABEND DESTRUCTIVE_OVERLAP SECOND_JVM_PROGRAM SECOND_H8_PROGRAM JVMPOOL_DISABLED JVM_PROFILE_NOT_FOUND JVM_PROFILE_NOT_VALID SYSTEM_PROPERTIES_NOT_FND USER_CLASS_NOT_FOUND
INVALID	INVALID_FUNCTION

PGPG gate, INITIAL_LINK function

The INITIAL_LINK function of the PGPG gate is used to link to the first program of a transaction.

Input parameters

PROGRAM_NAME is the name of the program being linked.

Output parameters

[ABEND_CODE] is the four-character abend code to be issued if there is an exception response with reason TRANSACTION_ABEND.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	AUTOINSTALL_FAILED AUTOINSTALL_INVALID_DATA AUTOINSTALL_MODEL_NOT_DEF AUTOINSTALL_URM_FAILED DESTRUCTIVE_OVERLAP INVALID_TERMINAL_TYPE PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE PROGRAM_NOT_DEFINED REMOTE_PROGRAM TRANSACTION_ABEND SECOND_JVM_PROGRAM SECOND_H8_PROGRAM JVMPOOL_DISABLED JVM_PROFILE_NOT_FOUND JVM_PROFILE_NOT_VALID SYSTEM_PROPERTIES_NOT_FND USER_CLASS_NOT_FOUND

PGRE gate, PREPARE_RETURN_EXEC function

The PREPARE_RETURN_EXEC function of the PGRE gate is used to process the communications area, inputmsg data, and transaction identifier from a user EXEC CICS RETURN command.

Input parameters

[TRANSID] is the four-character transaction identifier.

[COMMAREA] is the optional communications area made available to the linked program.

[INPUTMSG] is a data area to be supplied to the linked program on its first execution of an EXEC CICS RECEIVE command.

[IMMEDIATE] Indicates whether or not the transaction specified in TRANSID is to be attached as the next transaction regardless of any other transactions enqueued by ATI for this terminal. It can have either of these values:

YES|NO

[ENDACTIVITY]

indicates that a BTS activity is to be ended.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_COMMAREA_ADDR INVALID_COMMAREA_LEN INVALID_INPUTMSG_LEN INVALID_TERMINAL_TYPE INVALID_REQUEST_FROM_EXIT INVALID_RETURN_REQUEST NOT_INITIALIZED NO_TERMINAL TRANSID_NO_TERMINAL

PGXE gate, PREPARE_XCTL_EXEC function

The PREPARE_XCTL_EXEC function of the PGXE gate is used to process the communications area, inputmsg data, and transaction identifier from a user EXEC CICS XCTL command.

Input parameters

PROGRAM_NAME is the name of the program to which control is to be passed.

[COMMAREA] is the optional communications area made available to the linked program.

[INPUTMSG] is a data area to be supplied to the linked program on its first execution of an EXEC CICS RECEIVE command.

[SYSEIB_REQUEST]
specifies whether the EXEC CICS LINK had the SYSEIB translator option specified.

[SECURITY] indicates whether Program Manager must check security authorisation for the target program

Output parameters

[ABEND_CODE] is the four-character abend code to be issued if CICS drives the system default, which is to abend the transaction.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	AUTOINSTALL_FAILED AUTOINSTALL_INVALID_DATA AUTOINSTALL_MODEL_NOT_DEF AUTOINSTALL_URM_FAILED DESTRUCTIVE_OVERLAP INVALID_COMMAREA_ADDR INVALID_COMMAREA_LEN INVALID_INPUTMSG_LEN INVALID_TERMINAL_TYPE INVALID_REQUEST_FROM_EXIT NOT_INITIALIZED PROGRAM_NOT_AUTHORIZED PROGRAM_NOT_DEFINED PROGRAM_NOT_ENABLED PROGRAM_NOT_LOADABLE REMOTE_PROGRAM TRANSACTION_ABEND

PGXM gate, INITIALIZE_TRANSACTION function

The INITIALIZE_TRANSACTION function of the PGXM gate is used to initialize a transaction, and set up storage for the transaction.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION

PGXM gate, TERMINATE_TRANSACTION function

The TERMINATE_TRANSACTION function of the PGXM gate is used to terminate a transaction, and clean up the transaction-related storage at task termination.

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION

Program manager domain's generic gates

Table 80 summarizes the program manager domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 80. Program manager domain's generic gates

Gate	Trace	Function	Format
PGDM	PG 0101 PG 0102	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
PGST	PG 0F01 PG 0F02	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
PGUE	PG 1001 PG 1002	SET_EXIT_STATUS	APUE

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format APUE—"Application domain's generic formats" on page 593
Format DMDM—"Domain manager domain's generic formats" on page 669
Format STST—"Statistics domain's generic format" on page 1198

Initialize domain

There are two phases to initialization of the program manager domain:

1. The DFHPGDM module creates the PG domain anchor block, the PPT directory, and the PG Lock. It also adds subpools and gates, determines whether a cold, warm, or emergency start is needed, and waits for the global catalog to be available.
2. For a warm or emergency start, the DFHPGDM module rebuilds the PPT and restores the program autoinstall system initialization parameters from the global catalog entries. (It calls the parameter manager to obtain other system initialization parameter values.)

   For a cold start, the DFHPGDM module purges all the PPT entries from the global catalog.

Quiesce domain

In quiesce processing, the program manager domain:

- Sets the PG state to quiescing.
- Ensures that the statistics domain has gathered the PG statistics by issuing a WAIT_PHASE for STATISTICS_UNAVAILABLE.

  This also ensures synchronization with the AP domain quiesce activity.

- Does ***not*** delete PG gates; PG functions remain available. However, use of programs after this point does not appear in statistics. (DFHSTP issues a PC LINK/ PGLK LINK to DFHWKP after AP domain waits for STATISTICS_UNAVAILABLE).
- Does ***not*** write PPT entries to the global catalog. (PPT entries are only written to the catalog when they are installed or changed.)
- (Finally) Sets the PG state to quiesced.

Terminate domain

In terminate processing, the program manager domain sets the PG state to terminated, and makes the program manager domain unavailable to EXEC CICS commands.

Modules

Module	Function
DFHPGAI	A kernel subroutine called internally from the Program Manager to support the autoinstall for programs function.
DFHPGAQ	Handles the following requests: INQUIRE_AUTOINSTALL SET_AUTOINSTALL
DFHPGDD	Handles the following requests: DEFINE_PROGRAM DELETE_PROGRAM
DFHPGDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHPGDUF	PG domain offline dump formatting routine
DFHPGEX	Handles the following requests: INITIALIZE_EXIT TERMINATE_EXIT
DFHPGHM	Handles the following requests: SET_CONDITIONS IGNORE_CONDITIONS INQ_CONDITION SET_AIDS INQ_AID SET_ABEND INQ_ABEND PUSH_HANDLE POP_HANDLE FREE_HANDLE_TABLES CLEAR_LABELS
DFHPGIS	Handles the following requests: INQUIRE_PROGRAM INQUIRE_CURRENT_PROGRAM SET_PROGRAM START_BROWSE_PROGRAM GET_NEXT_PROGRAM END_BROWSE_PROGRAM REFRESH_PROGRAM
DFHPGLD	Handles the following requests: LOAD_EXEC LOAD RELEASE_EXEC RELEASE
DFHPGLE	Handles the following requests: LINK_EXEC

Module	Function
DFHPGLK	Handles the following requests: LINK LINK_PLT
DFHPGLU	Handles the following requests: LINK_URM
DFHPGPG	Handles the following requests: INITIAL_LINK
DFHPGRE	Handles the following requests: PREPARE_RETURN_EXEC
DFHPGRP	Program manager domain recovery program, responsible for recovering program definitions from the global catalog.
DFHPGST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHPGUE	Handles program manager domain service requests.
DFHPGTRI	Interprets PG domain trace entries
DFHPGXE	Handles the following requests: PREPARE_XCTL_EXEC
DFHPGXM	Handles the following requests: INITIALIZE_TRANSACTION TERMINATE_TRANSACTION

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the program manager domain are of the form PG xxxx; the corresponding trace levels are PG 1, PG 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 96. Pipeline Manager Domain (PI)

Pipeline Manager Domain's specific gates

Table 81 summarizes the Pipeline Manager domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 81. Pipeline Manager domain's specific gates

Gate	Trace	Function	XPI
PIAT	PI 0D00 PI 0D01	CREATE_CONTEXT CREATE_CONTEXT_RESP CREATE_NON_TERMINAL_MSG CREATE_REGISTER_REQUEST CREATE_REGISTER_RESP CREATE_TERMINAL_MSG PROCESS_CONTEXT PROCESS_CONTEXT_RESP PROCESS_MSG	No
PICC	PI 0F30 PI 0F31	FIND_SIGNATURE HANDLE_PARSE_EVENT PERFORM_XML_PARSE	No
PIII	PI 0F00 PI 0F01	PARSE_ICM	No
PIPL	PI 0400 PI 0401	ADD_PIPELINE COMPLETE_PIPELINE DISCARD_PIPELINE END_BROWSE_PIPELINE ESTABLISH_PIPELINE GET_NEXT_PIPELINE INQUIRE_PIPELINE PERFORM_PIPELINE RELINQUISH_PIPELINE RESOLVE_PIPELINE SET_PIPELINE START_BROWSE_PIPELINE	No
PIPM	PI 0A00 PI 0A01	INVOKE_PROGRAM INVOKE_STUB START_PIPELINE	No
PIRE	PI 0D90 PI 0D92	PERFORM_RESYNC	No
PISC	PI 0E00 PI 0E01	DYN_CREATE_WEBSERVICE UPDATE_WEBSERVICE	No
PISF	PI 0C00 PI 0C01	SOAPFAULT_ADD SOAPFAULT_CREATE SOAPFAULT_DELETE	No
PISN	PI 0C10 PI 0C11	SOAP_11 SOAP_12	No
PITG	PI 0700 PI 0701	SEND_REQUEST SEND_RESPONSE CONVERSE RECEIVE_REQUEST SEND_ERROR_RESPONSE	No

Table 81. Pipeline Manager domain's specific gates (continued)

Gate	Trace	Function	XPI
PITL	PI 1000 PI 1001	PROCESS_SOAP_REQUEST_FUNCTION	No
PIWR	PI 0300 PI 0301	CREATE_WEBSERVICE DECREMENT_USE_COUNT DISCARD_WEBSERVICE END_BROWSE_WEBSERVICE GET_NEXT_WEBSERVICE INCREMENT_USE_COUNT INITIALISE_WEBSERVICE INQUIRE_WEBSERVICE RESOLVE_ALL_WEBSERVICES SET_WEBSERVICE START_BROWSE_WEBSERVICE	No

PIAT gate, CREATE_CONTEXT function

Creates a WSAT coordination context SOAP header.

Input Parameters

POOL_TOKEN
A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
- ABEND
- INVALID_FORMAT
- INVALID_FUNCTION
- LOOP
- NO_CHANNEL
- PGCR_GET_ERROR
- PGCR_PUT_ERROR
- SMGF_ERROR
- TASK_CANCELLED
- TIMED_OUT

RESPONSE
Values for the parameter are:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

PIAT gate, CREATE_CONTEXT_RESP function

Create a null context response, which is returned when a WSAT participant send back its output.

Input Parameters

POOL_TOKEN
A token to the current container pool, which holds data used to build the header, and where the populated dfhheader container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIAT gate, CREATE_NON_TERMINAL_MSG function

Create a non-terminal SOAP message used in WS-AtomicTransaction two-phase commit protocol processing. Non-terminal messages anticipate a response. They are used to convey the following function requests: Prepare, Commit, Rollback, and Replay.

Input Parameters

NOTIFICATION_TYPE

Values for the parameter are:

COMMIT
PREPARE
ROLLBACK

POOL_TOKEN

A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Values for the parameter are:

OK
EXCEPTION

DISASTER
INVALID
KERNERROR
PURGED

PIAT gate, CREATE_REGISTER_REQUEST function

Create a WSAT registration request SOAP message.

Input Parameters

POOL_TOKEN
A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIAT gate, CREATE_REGISTER_RESP function

Create a WSAT registration response SOAP message.

Input Parameters

POOL_TOKEN
A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR

TASK_CANCELLED
TIMED_OUT

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIAT gate, CREATE_TERMINAL_MSG function

Create a terminal SOAP message used in WS-AtomicTransaction two-phase commit protocol processing. Terminal messages do not anticipate a response. They are used to convey the following function requests: Prepared, Committed, Aborted, and Readonly.

Input Parameters

NOTIFICATION_TYPE
Values for the parameter are:
ABORTED
COMMITTED
PREPARED
READONLY

POOL_TOKEN
A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIAT gate, PROCESS_CONTEXT function

Process a WS-AtomicTransaction coordination context header.

Input Parameters

POOL_TOKEN
A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
- ABEND
- INVALID_FORMAT
- INVALID_FUNCTION
- LOOP
- NO_CHANNEL
- PGCR_GET_ERROR
- PGCR_PUT_ERROR
- SMGF_ERROR
- TASK_CANCELLED
- TIMED_OUT

RESPONSE

Values for the parameter are:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

PIAT gate, PROCESS_CONTEXT_RESP function

Input Parameters

POOL_TOKEN

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
- ABEND
- INVALID_FORMAT
- INVALID_FUNCTION
- LOOP
- NO_CHANNEL
- PGCR_GET_ERROR
- PGCR_PUT_ERROR
- SMGF_ERROR
- TASK_CANCELLED
- TIMED_OUT

RESPONSE

Values for the parameter are:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

PIAT gate, PROCESS_MSG function

Process a WS-AtomicTransaction message. This can be a Register Request, a Register Response, a Non Terminal message, or a Terminal Message.

Input Parameters

POOL_TOKEN
A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
- ABEND
- INVALID_FORMAT
- INVALID_FUNCTION
- LOOP
- NO_CHANNEL
- PGCR_GET_ERROR
- PGCR_PUT_ERROR
- SMGF_ERROR
- TASK_CANCELLED
- TIMED_OUT

RESPONSE
Values for the parameter are:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

PICC gate, FIND_SIGNATURE function

Determine an operation from its signature

Input Parameters

OUTPUT_DATA
A pointer to the operation in the internal COMMAREA or container model (ICM)

XML_BODY_STRING
The incoming SOAP message

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
- HEAP_INIT_FAILURE
- INSUFFICIENT_STORAGE
- INTERNAL_FAILURE
- INVALID_PARSE_STATE
- SAXHANDLER_LINK_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
- FIXED_ELEMENT_COUNT
- HEAP_ALLOCATE_FAILURE
- HEAP_RELEASE_FAILURE
- ICM_ENTRY_NOT_FOUND
- INQUIRE_CHANNEL_FAILED

OUTPUT_BUFFER_OVERFLOW
PUT_CONTAINER_FAILED
SOAP_FAULT

The following values are returned when RESPONSE is EXCEPTION:
COMMAREA_LENGTH
INVALID_FUNCTION
INVALID_ICM_TYPE
INVALID_INPUT

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PICC gate, HANDLE_PARSE_EVENT function

Handle an XML parse event when located by the PL/I SAX parser

Input Parameters

EVENT_TOKEN
A pointer to the event token provided by the XML parser.

EVENT_TOKEN_LENGTH
The length of the event token.

EVENT_TYPE
A BIN(31) value indicating what event has been signaled by the parser.

HANDLER_WORK_TOKEN
A pointer to the DFHPICC work area.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
HEAP_INIT_FAILURE
INSUFFICIENT_STORAGE
INTERNAL_FAILURE
INVALID_PARSE_STATE
SAXHANDLER_LINK_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
FIXED_ELEMENT_COUNT
HEAP_ALLOCATE_FAILURE
HEAP_RELEASE_FAILURE
ICM_ENTRY_NOT_FOUND
INQUIRE_CHANNEL_FAILED
OUTPUT_BUFFER_OVERFLOW
PUT_CONTAINER_FAILED
SOAP_FAULT

The following values are returned when RESPONSE is INVALID:
COMMAREA_LENGTH
INVALID_FUNCTION
INVALID_INPUT

RESPONSE
Values for the parameter are:
OK

EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PICC gate, PERFORM_XML_PARSE function

Parse a SOAP body and convert the data elements into a COMMAREA format.

Input Parameters

ICM_ADDRESS
The address of the internal COMMAREA or container model (ICM) which is to be used for the SOAP to COMMAREA conversion.

OUTPUT_DATA
A pointer to, and length of, the COMMAREA into which the SOAP body has been mapped.

XML_BODY_STRING
A pointer to the incoming SOAP body.

CHANNEL_NAME
The name of the channel which contains the SOAP body.

XML_HEADER_NS
Optional Parameter

A pointer to the XML namespace information for the SOAP body.

XML_OPERATION
Optional Parameter

The operation name for which the SOAP body is intended.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
HEAP_INIT_FAILURE
INSUFFICIENT_STORAGE
INTERNAL_FAILURE
INVALID_PARSE_STATE
SAXHANDLER_LINK_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
FIXED_ELEMENT_COUNT
HEAP_ALLOCATE_FAILURE
HEAP_RELEASE_FAILURE
ICM_ENTRY_NOT_FOUND
INQUIRE_CHANNEL_FAILED
OUTPUT_BUFFER_OVERFLOW
PUT_CONTAINER_FAILED
SOAP_FAULT

The following values are returned when RESPONSE is INVALID:
COMMAREA_LENGTH
INVALID_FUNCTION
INVALID_INPUT

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID

KERNERROR
PURGED

PIII gate, PARSE_ICM function

Convert an outbound COMMAREA or container into a SOAP body.

Input Parameters

CHANNEL_NAME
Optional parameter

The name of the channel which holds the container with the SOAP body.

INPUT_COMMAREA
The address and length of the COMMAREA or container to convert.

OUTPUT_ICM_ADDRESS
The address of the internal COMMAREA or container model (ICM) that defines how to map the COMMAREA or container to a SOAP body.

OUTPUT_XML
The address of the SOAP body.

Output Parameters

REASON
Values for the parameter are:
ABEND
BUFFER_OVERFLOW
CONTAINER_GET_FAILURE
FREEMAIN_FAILURE
GETMAIN_FAILURE
HEAP_INIT_FAILURE
ICM_NOT_FOUND
INPUT_ERROR
INSUFFICIENT_STORAGE
INVALID_FORMAT
INVALID_FUNCTION
INVALID_ICM_DATATYPE
MALLOC_FAILURE
NOT_AUTHORIZED
RELEASE_FAILURE
SEVERE_ERROR

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PIIW gate, INVOKE_WEBSERVICE function

This function supports the INVOKE WEBSERVICE API where CICS is acting as Web Service Requester. Depending upon the attributes specified in the WEBSERVICE resource, it calls the Pipeline Manager (DFHPIPM) to start the pipeline, or it links directly to an application program directly.

Input Parameters

CHANNEL
The name of a channel which holds the container in which data is passed to the target WEBSERVICE.

OPERATION
The name of the operation which is to be invoked.

WEBSERVICE
The name of the WEBSERVICE resource.

URI
Optional Parameter

The URI of the target Web service. If this parameter is omitted, the WEBSERVICE resource must specify an endpoint or a program.

Output Parameters

REASON
Values for the parameter are:

```
CHANNEL_NOT_FOUND
CHANNEL_ERROR
CONTAINER_DATATYPE_ERR
CONTAINER_NOT_FOUND
ENDPOINT_NOT_PROVIDED
INVALID_CHANNEL_NAME
INVALID_FUNCTION
INVALID_OPERATION
INVALID_URI
INVALID_WSBIND_FORMAT
OPERATION_NOT_FOUND
PARSE_CONVERSION_ERROR
PARSE_INPUT_ERROR
PIPELINE_MODE_MISMATCH
PIPELINE_NOT_ACTIVE
PIPELINE_NOT_FOUND
PROGRAM_LINK_FAILED
SOAP_FAULT_BUILT
UNHANDLED_PIPELINE_ERROR
VENDOR_LINK_FAILED
WEBSERVICE_NOT_FOUND
WEBSERVICE_NOT_INSERVICE
```

RESPONSE
Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

SOAP_FAULT_RESP1
The response that was returned from the SOAP message handler's fault processing in the DFHWS-RESPCODES container.

SOAP_FAULT_RESP2
The reason that was returned from the SOAP message handler's fault processing in the DFHWS-RESPCODES container.

PIPL gate, ADD_PIPELINE function

Add a PIPELINE definition to the system.

Input Parameters

CONFIGFILE
The fully qualified name of the XML pipeline configuration file on HFS.

PIPELINE
The name of the PIPELINE.

SHELF
The fully qualified name of a directory (or shelf) primarily for WSBIND and WSDL files.

STATUS
The initial state of the PIPELINE.

Values for the parameter are:

DISABLED
ENABLED

WSDIR
Optional Parameter

The fully qualified name of the WSBIND directory on HFS.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

CATALOG_ERROR
DIRECTORY_ERROR
INVALID_HFSNAME
INVALID_NAME
INVALID_SHELF
INVALID_STATUS
INVALID_WSDIR
NOT_AUTHORIZED
NOT_DISABLED
WSDIR_INACCESIBLE

RESPONSE
Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIPL gate, COMPLETE_PIPELINE function

Complete the installation of a PIPELINE. PIPELINEs are installed in two phases: this is the second, called after CICS initialization is complete. This function reads data from the files in HFS and builds the internal control blocks.

Input Parameters

PIPELINE
The name of the PIPELINE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

CATALOG_ERROR
DIRECTORY_ERROR
INVALID_HFSNAME
INVALID_NAME
INVALID_SHELF
INVALID_STATUS
INVALID_WSDIR
NOT_AUTHORIZED
NOT_DISABLED

RESPONSE
: Values for the parameter are:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

PIPL gate, DISCARD_PIPELINE function

Discard a PIPELINE.

Input Parameters

PIPELINE
: The name of the PIPELINE.

Output Parameters

REASON
: The following values are returned when RESPONSE is EXCEPTION:
  - CATALOG_ERROR
  - DISCARD_IN_PROGRESS
  - INVALID_BROWSE_TOKEN
  - NOT_AUTHORIZED
  - NOT_DISABLED
  - NOT_FOUND

RESPONSE
: Values for the parameter are:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

PIPL gate, END_BROWSE_PIPELINE function

End the browse operation on the PIPELINE resources that are installed in the system.

Input Parameters

BROWSETOKEN
: A token that represents the browse operation on subsequent GET_NEXT_PIPLINE and END_BROWSE requests.

Output Parameters

REASON
: The following values are returned when RESPONSE is EXCEPTION:
  - ABEND
  - INVALID_BROWSE_TOKEN
  - LOOP

RESPONSE
: Values for the parameter are:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR

PURGED

PIPL gate, ESTABLISH_PIPELINE function

Check that a PIPELINE is in a state in which it can be used, and increment its use count.

Input Parameters

PIPELINE
The name of the PIPELINE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CATALOG_ERROR
INVALID_STATUS
NOT_AUTHORIZED
NOT_FOUND

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIPL gate, GET_NEXT_PIPELINE function

During a browse operation, extract information about the next PIPELINE.

Input Parameters

BROWSETOKEN
The browse token that was returned by the START_BROWSE_PIPELINE function.

CONFIGFILE_BUFF
Optional Parameter

A buffer in which the fully qualified name of the XML pipeline configuration file on HFS is returned.

RESET
Optional Parameter

A parameter indicating whether the statistics for the PIPELINE are to be reset.

Values for the parameter are:
NO
YES

SHELF_BUFF
Optional Parameter

A buffer in which the fully qualified name of the directory (or shelf) for WSBIND and WSDL files is returned.

WSDIR_BUFF
Optional Parameter

A buffer in which the fully qualified name of the WSBIND directory on HFS is returned.

Output Parameters

PIPELINE
The name of the PIPELINE.

REASON
The following values are returned when RESPONSE is EXCEPTION:
- ABEND
- BROWSE_END
- INVALID_BROWSE_TOKEN
- LOCK_ERROR
- LOOP
- PARMS_STORAGE_ERROR
- SETUP_ERROR
- STORAGE_ERROR

RESPONSE
Values for the parameter are:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

STATUS
Optional Parameter

The current status of the PIPELINE.

Values for the parameter are:
- DISABLING
- DISABLED
- DISCARDING
- ENABLED
- ENABLING

TOTAL_USE_COUNT
Optional Parameter

The current use count of the PIPELINE.

PIPL gate, INQUIRE_PIPELINE function

Inquire on the attributes, state and associated resources of a PIPELINE.

Input Parameters

PIPELINE
The name of the PIPELINE.

CONFIGFILE_BUFF
Optional Parameter

A buffer in which the fully qualified name of the XML pipeline configuration file on HFS is returned.

DERIVED_SHELF_BUFF
Optional Parameter

A buffer in which the fully qualified name of the HFS file which contains the WSDL for the PIPELINE is returned.

SHELF_BUFF
Optional Parameter

A buffer in which the fully qualified name of the directory (or shelf) for WSBIND and WSDL files is returned.

WSDIR_BUFF
Optional Parameter

A buffer in which the fully qualified name of the WSBIND directory on HFS is returned.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NOT_AUTHORIZED
NOT_FOUND

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

MODE
Optional Parameter

The MODE of the PIPELINE.

Values for the parameter are:
PROVIDER
REQUESTER
UNKNOWN

PIPELINE_TOKEN
Optional Parameter

A token which can be used by other parts of the domain to refer to the PIPELINE.

STATUS
Optional Parameter

The current status of the PIPELINE.

Values for the parameter are:
DISABLING
DISABLED
DISCARDING
ENABLED
ENABLING

TOTAL_USE_COUNT
Optional Parameter

The current use count of the PIPELINE.

PIPL gate, PERFORM_PIPELINE function

Perform the specified action on a PIPELINE.

Input Parameters

ACTION
The only supported action is SCAN. The PIPELINE is scanned for WSBIND files which are then installed.

Values for the parameter are:
SCAN

PIPELINE
The name of the PIPELINE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND

DUPLICATE
INVALID_ACTION
INVALID_STATUS
LOOP
NOT_AUTHORIZED
NOT_FOUND
PIPELINE_SCAN_ERROR
SCAN_ALREADY_IN_PROGRESS
WSDIR_INACCESSIBLE

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIPL gate, RELINQUISH_PIPELINE function

Relinquish the use of a PIPELINE. The use count is decremented, and if it is then zero, and the PIPELINE's state is DISABLING, the status changes to DISABLED.

Input Parameters

PIPELINE
The name of the PIPELINE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CATALOG_ERROR
NOT_AUTHORIZED
NOT_FOUND

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIPL gate, RESOLVE_PIPELINE function

For each PIPELINE, start a transaction to complete PIPELINE installation. The function is used at the end of domain initialization.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
LOOP
SETUP_ERROR
STORAGE_ERROR

RESPONSE
Values for the parameter are:
OK

EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIPL gate, SET_PIPELINE function

Set a PIPELINE to DISABLED or ENABLED state.

Input Parameters

PIPELINE
The name of the PIPELINE.

STATUS
The state to be set.

Values for the parameter are:
DISABLED
ENABLED

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_STATE
NOT_AUTHORIZED
NOT_FOUND

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIPL gate, START_BROWSE_PIPELINE function

Start browsing the installed PIPELINE resources.

Input Parameters

PIPELINE
Optional Parameter

The name of the PIPELINE at which the browse is to begin.

Output Parameters

BROWSETOKEN
A token that identifies the browse operation to subsequent GET_NEXT_PIPELINE and END_BROWSE reqeusts.

REASON
Values for the parameter are:
ABEND
INVALID_PIPELINE
LOCK_ERROR
LOOP
SETUP_ERROR
STORAGE_ERROR

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIPM gate, INVOKE_PROGRAM function

Invoke a PIPELINE's application programs. The function can change the transaction's context, and the request can be routed to another region.

Input Parameters

CHANNEL
The channel to be passed to the target program.

PROGRAM
The program to be invoked.

APPLID
Optional Parameter

The APPLID to be used for the execution of the application program.

RS_PUBLIC_ID
Optional Parameter

The request stream public identifier to be associated with the transaction.

TRANSID
Optional Parameter

The transaction identifier to be used to execute the application program.

USERID
Optional Parameter

The user ID to be used for the execution of the application program.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:
CHANNEL_ERROR
CONTEXT_SWITCH_FAILED
NO_CHANNEL
PIPELINE_MODE_MISMATCH
PIPELINE_NOT_ACTIVE
PIPELINE_NOT_FOUND
RZ_CREATE_FAILURE
RZ_TRANSPORT_ERROR
TARGET_PROGRAM_UNAVAILABLE
UNHANDLED_NODE_FAILURE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:

TASK_CANCELLED
TIMED_OUT

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIPM gate, INVOKE_STUB function

Invoke an application program remotely.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:
CHANNEL_ERROR
CONTEXT_SWITCH_FAILED
NO_CHANNEL
PIPELINE_MODE_MISMATCH
PIPELINE_NOT_ACTIVE
PIPELINE_NOT_FOUND
RZ_CREATE_FAILURE
RZ_TRANSPORT_ERROR
TARGET_PROGRAM_UNAVAILABLE
UNHANDLED_NODE_FAILURE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PIPM gate, START_PIPELINE function

Start a requester or provider pipeline.

Input Parameters

MODE
Parameter indicating whether the pipeline is to be started for a service requester or for a service provider.

Values for the parameter are:

PROVIDER
REQUESTER

PIPELINE
The name of the PIPELINE resource.

CHANNEL
Optional Parameter

The name of a channel holding containers to be passed to the pipeline.

TRANSPORT_NAME
Optional Parameter

Depending upon the value of the TRANSPORT_TYPE parameter, the name of a TCPIPSERVICE or an MQ queue to be passed to the pipeline.

TRANSPORT_TYPE
Optional Parameter

Parameter indicating the type of transport.

Values for the parameter are:

HTTP
MQ

WEBSERVICE
Optional Parameter

The name of the WEBSERVICE to be invoked for this pipeline.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
LOCK_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:

CHANNEL_ERROR
CONTEXT_SWITCH_FAILED
NO_CHANNEL
PIPELINE_MODE_MISMATCH
PIPELINE_NOT_ACTIVE
PIPELINE_NOT_FOUND
RZ_CREATE_FAILURE
RZ_TRANSPORT_ERROR
TARGET_PROGRAM_UNAVAILABLE
UNHANDLED_NODE_FAILURE

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:

TASK_CANCELLED
TIMED_OUT

RESPONSE
Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR

PURGED

PIRE gate, PERFORM_RESYNC function

Resynchronize any WS-AtomicTransaction units of work that are in-doubt, following a restart of CICS.

Input Parameters

None.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ALREADY_IN_RESYNC

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PISC gate, DYN_CREATE_WEBSERVICE function

This function dynamically creates a WEBSERVICE resource via a PIPELINE scan.

Input Parameters

PIPELINE
The name of the PIPELINE resource that owns the WEBSERVICE.

WSBIND
The fully qualified location of the Web service binding file in the pickup directory in the hierarchical file system (HFS).

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CREATE_FAILED
DISCARD_FAILED
INQUIRE_FAILED
INQUIRE_HFS_FAILED
NAME_CLASH
NO_UPDATE_NEEDED
UPDATE_PENDING
WSDL_NAME_TOO_LONG

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PISC gate, UPDATE_WEBSERVICE function

This function completes the updating of a WEBSERVICE resource. It is invoked when the use count for a WEBSERVICE which is in UPDATING state reaches zero.

Input Parameters

WEBSERVICE
The name of the WEBSERVICE whose update is to be completed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
- CREATE_FAILED
- DISCARD_FAILED
- INQUIRE_FAILED
- INQUIRE_HFS_FAILED
- NAME_CLASH
- NO_UPDATE_NEEDED
- UPDATE_PENDING
- WSDL_NAME_TOO_LONG

RESPONSE
Values for the parameter are:
- OK
- EXCEPTION
- DISASTER
- PURGED

PISF gate, SOAPFAULT_ADD function

Add extra data to a SOAP fault created by the SOAPFAULT_CREATE function.

Input Parameters

FAULT_STRING
The description of the fault in a readable form.

SUBCODE_STRING
The value to put in the `<subcode>` element of a SOAP fault.

CCSID
Optional Parameter

The CCSID of the input.

NATLANG
Optional Parameter

The `xml:lang` value for the FAULT_STRING

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
- CCSID_CONVERSION_ERROR
- CCSID_INVALID
- CCSID_PARTIAL_CONVERSION
- CCSID_UNSUPPORTED
- INVALID_CODE
- INVALID_REQUEST
- NO_FAULT
- SEVERE_ERROR

RESPONSE

Values for the parameter are:
- OK
- EXCEPTION
- DISASTER
- INVALID

KERNERROR
PURGED

PISF gate, SOAPFAULT_CREATE function

Create a SOAP fault in an internal format.

Input Parameters

FAULT_STRING
The description of the fault in a readable form.

FAULTCODE
The standard SOAP fault code to use

FAULTCODE_STRING
The value to use for the `<faultcode>` element instead of a standard one.

CCSID
Optional Parameter

The CCSID of the input.

DETAIL
Optional Parameter

XML containing detailed fault data.

FAULT_ACTOR
Optional Parameter

The value to put in the `<faultactor>` element.

NATLANG
Optional Parameter

The `xml:lang` value for the FAULT_STRING parameter.

ROLE
Optional Parameter

The value to put in the `<role>` element.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CCSID_CONVERSION_ERROR
CCSID_INVALID
CCSID_PARTIAL_CONVERSION
CCSID_UNSUPPORTED
INVALID_CODE
INVALID_REQUEST
SEVERE_ERROR

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PISF gate, SOAPFAULT_DELETE function

Delete the internal form of a SOAP fault.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NO_FAULT
NOT_FOUND
SEVERE_ERROR

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

PISN gate, SOAP_11 function

Start a message handler to process SOAP 1.1 messages.

Output Parameters

SOAPFAULT
indicates whether a SOAP fault has been built.

Values for the parameter are:
NONE
FAULT_BUILT

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
BAD_FAULT
SEVERE_ERROR

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PISN gate, SOAP_12 function

Start a message handler to process SOAP 1.2 messages.

Output Parameters

SOAPFAULT
indicates whether a SOAP fault has been built.

Values for the parameter are:
NONE
FAULT_BUILT

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
BAD_FAULT
SEVERE_ERROR

RESPONSE
Values for the parameter are:
OK
EXCEPTION

DISASTER
PURGED

PITG gate, SEND_REQUEST function

Input Parameters

None

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
invalid_codepage
socket_error
unknown_host

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
MQ_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
INVALID_PARAMETER

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PITG gate, SEND_RESPONSE function

Input Parameters

None

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
invalid_codepage
socket_error
unknown_host

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
MQ_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
INVALID_PARAMETER

RESPONSE
: Values for the parameter are:
    OK
    EXCEPTION
    DISASTER
    PURGED

PITG gate, CONVERSE function

Input Parameters

None

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`invalid_codepage`
`socket_error`
`unknown_host`

The following values are returned when RESPONSE is INVALID:
`INVALID_FORMAT`
`INVALID_FUNCTION`

The following values are returned when RESPONSE is DISASTER:
`ABEND`
`MQ_FAILURE`

The following values are returned when RESPONSE is EXCEPTION:
`INSUFFICIENT_STORAGE`
`INVALID_PARAMETER`

RESPONSE
Values for the parameter are:
`OK`
`EXCEPTION`
`DISASTER`
`PURGED`

RECEIVE_REQUEST

Input Parameters

None

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`codepage_not_found`
`connection_closed`
`socket_error`

The following values are returned when RESPONSE is INVALID:
`INVALID_FORMAT`
`INVALID_FUNCTION`

The following values are returned when RESPONSE is DISASTER:
`ABEND`
`MQ_FAILURE`

The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
INVALID_PARAMETER

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PITG gate, SEND_ERROR_RESPONSE function

Input Parameters

None

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
MQ_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
INVALID_PARAMETER

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PITL gate, PROCESS_SOAP_REQUEST function

Process a SOAP body received on a SOAP pipeline

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
APP_FAULT
CONV_FROM_SOAP_FAILED
CONV_TO_SOAP_FAILED
INBOUND_VALIDATION_FAILED
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NOT_AUTHORIZED
OPERATION_NOT_FOUND
OUTBOUND_VALIDATION_FAILED
SEVERE_ERROR
SOAP_BODY_CONTAINER_FAULT
TARGET_ABENDED

TARGET_LINK_FAILED
VENDOR_LINK_FAILED
WSBIND_FORMAT_INVALID

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PIWR gate, CREATE_WEBSERVICE function

Create a new WEBSERVICE resource.

Input Parameters

PIPELINE
The pipeline which will own the WEBSERVICE.

WEBSERVICE
The name of the WEBSERVICE.

WSBIND_BUF
The location of the Web service binding file in the hierarchical file system (HFS).

SCAN_MODE
Optional Parameter

Indicates whether the WEBSERVICE is being scanned in or not.

Values for the parameter are:
NO
YES

VALIDATION
Optional Parameter

Indicates whether validation is enabled for the WEBSERVICE.

Values for the parameter are:
NO
YES

WARM_RESTART
Optional Parameter

Indicates whether the WEBSERVICE is to be recovered from the catalog during a warm restart.

Values for the parameter are:
NO
YES

WSDLFILE_BUF
Optional Parameter

The location of the optional Web service description (WSDL) file in the hierarchical file system (HFS).

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
DIRECTORY_ERROR
INSUFFICIENT_STORAGE
LOCK_FAILURE
PIPELINE_ERROR
PIPELINE_NON_EXISTANT
SEVERE_ERROR

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PIWR gate, DECREMENT_USE_COUNT function

Decrement the current use count for a WEBSERVICE. When it reaches 0 and if the WEBSERVICE is updating or discarding then the completion of the update or discard operation will be triggered.

Input Parameters

WEBSERVICE
The name of the WEBSERVICE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
SEVERE_ERROR

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PIWR gate, DISCARD_WEBSERVICE function

This function discards a WEBSERVICE resource.

Input Parameters

WEBSERVICE
The name of the WEBSERVICE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
NOT_AUTHORIZED
SEVERE_ERROR
WEBSERVICE_IN_USE
WEBSERVICE_NOT_FOUND

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PIWR gate, END_BROWSE_WEBSERVICE function

This function ends a browse operation for WEBSERVICE resources.

Input Parameters

BROWSE_TOKEN
The browse token for the browse operation.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`INVALID_BROWSE_TOKEN`

RESPONSE
Values for the parameter are:
`OK`
`EXCEPTION`
`DISASTER`
`INVALID`
`KERNERROR`
`PURGED`

PIWR gate, GET_NEXT_WEBSERVICE function

Get the next WEBSERVICE resource during a browse operation.

Input Parameters

BROWSE_TOKEN
The browse token for the browse operation.

BINDING_BUF
Optional Parameter

A buffer in which the WSDL binding value is returned.

ENDPOINT_BUF
Optional Parameter

A buffer in which the end point URI is returned.

RESET
Optional Parameter

A flag that indicates if the use count is to be reset to zero.

Values for the parameter are:
`NO`
`YES`

WSBIND_BUF
Optional Parameter

A buffer in which the location of the Webservice binding file in the hierarchical file system (HFS) is returned.

WSDLFILE_BUF
Optional Parameter

A buffer in which the location of the Web service description (WSDL) file in the hierarchical file system (HFS) is returned.

Output Parameters

DATESTAMP
The date stamp of the Web service binding file

LASTMODTIME
The time at which the Web service binding file was last changed.

PGMINTERFACE
The type of interface used by the target program

Values for the parameter are:

CHANNEL
COMMAREA

PIPELINE
The pipeline which owns the WEBSERVICE.

PROGRAM
The target program.

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
INVALID_BROWSE_TOKEN

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

STATE
The current state of the WEBSERVICE.

Values for the parameter are:
DISCARDING
INITING
INSERVICE
UNUSABLE
UPDATING

TIMESTAMP
The time stamp of the Web service binding file.

URIMAP
The name of the URIMAP that is associated with the WEBSERVICE.

VALIDATION
Indicates whether validation is enabled for the WEBSERVICE.

Values for the parameter are:
NO
YES

WEBSERVICE
The name of the WEBSERVICE.

TOTAL_USE_COUNT
Optional Parameter

The current use count for the WEBSERVICE.

PIWR gate, INCREMENT_USE_COUNT function

Increment the use count for the named WEBSERVICE.

Input Parameters

WEBSERVICE
The name of the WEBSERVICE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
SEVERE_ERROR

RESPONSE
: Values for the parameter are:
    - OK
    - EXCEPTION
    - DISASTER
    - PURGED

PIWR gate, INITIALISE_WEBSERVICE function

Resolve the HFS parts of a WEBSERVICE. The function takes a WEBSERVICE which is in INSTALLING state to either INSERVICE or UNUSABLE state.

Input Parameters

WEBSERVICE
: The name of the WEBSERVICE.

Output Parameters

REASON
: The following values are returned when RESPONSE is EXCEPTION:
    - ABEND
    - EYECATCHER_ERROR
    - FILE_NOT_FOUND
    - INSUFFICIENT_STORAGE
    - NOT_AUTHORIZED
    - PIPELINE_ERROR
    - PIPELINE_WRONG_MODE
    - READ_ERROR
    - SEVERE_ERROR
    - SHELF_WRITE_ERROR
    - VERSION_ERROR
    - WEBSERVICE_NOT_FOUND

RESPONSE
: Values for the parameter are:
    - OK
    - EXCEPTION
    - DISASTER
    - PURGED

PIWR gate, INQUIRE_WEBSERVICE function

Inquire on a WEBSERVICE resource.

Input Parameters

WEBSERVICE
: The name of the WEBSERVICE.

BINDING_BUF
: Optional Parameter

  A buffer in which the WSDL binding value is returned.

ENDPOINT_BUF
: Optional Parameter

  A buffer in which the endpoint URI is returned.

WSBIND_BUF
: Optional Parameter

  A buffer in which the location of the Web service binding file in the hierarchical file system (HFS) is returned.

WSDLFILE_BUF
Optional Parameter

A buffer in which the location of the optional Web service description (WSDL) file in the hierarchical file system (HFS) is returned.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
- ABEND
- NOT_AUTHORIZED
- SEVERE_ERROR
- WEBSERVICE_NOT_FOUND

RESPONSE
Values for the parameter are:
- OK
- EXCEPTION
- DISASTER
- PURGED

CONTAINER
Optional Parameter

The name of the container for the target program's data.

DATESTAMP
Optional Parameter

The date stamp of the Web service binding file.

LASTMODTIME
Optional Parameter

The time at which the Web service binding file was last changed.

PGMINTERFACE
Optional Parameter

The type of interface used by the target program

Values for the parameter are:
- CHANNEL
- COMMAREA
- NOTAPPLIC

PGMINTERFACE
The type of interface used by the target program

Values for the parameter are:
- CHANNEL
- COMMAREA
- NOTAPPLIC

PIPELINE
Optional Parameter

The pipeline which owns the WEBSERVICE.

PROGRAM
Optional Parameter

The target program.

STATE
Optional Parameter

The current state of the WEBSERVICE.

Values for the parameter are:

DISCARDING
INITING
INSERVICE
UNUSABLE
UPDATING

TIMESTAMP
Optional Parameter

The time stamp of the Web service binding file.

TOTAL_USE_COUNT
Optional Parameter

The total use count for the WEBSERVICE.

URIMAP
Optional Parameter

The name of the URIMAP that is associated with the WEBSERVICE.

VALIDATION
Optional Parameter

Indicates whether validation is enabled for the WEBSERVICE.

Values for the parameter are:
NO
YES

WSADDR
Optional Parameter

The address of the WEBSERVICE control block.

PIWR gate, RESOLVE_ALL_WEBSERVICES function

Resolve all WEBSERVICE resources for a given pipeline that are in INITING state.

Input Parameters

PIPELINE
Optional Parameter

The pipeline for which WEBSERVICE resources are to be resolved.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
SEVERE_ERROR

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PIWR gate, SET_WEBSERVICE function

Change the state of a WEBSERVICE resource.

Input Parameters

VALIDATION
The new validation state for the WEBSERVICE.

Values for the parameter are:

NO
YES

WEBSERVICE
The name of the WEBSERVICE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
DUPLICATE
NOT_AUTHORIZED
SEVERE_ERROR
WEBSERVICE_NOT_FOUND

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

PIWR gate, START_BROWSE_WEBSERVICE function

Start a browse operation on WEBSERVICE resources.

Output Parameters

BROWSE_TOKEN
The browse token for the browse operation.

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
BROWSE_END
DIRECTORY_ERROR
DUPLICATE
FILE_NOT_FOUND
FREEMAIN_FAILURE
INSUFFICIENT_STORAGE
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
LOCK_FAILURE
LOOP
NO_WEBS_INSTALLED
NOT_AUTHORIZED
PIPELINE_ERROR
PIPELINE_NON_EXISTANT
PIPELINE_WRONG_MODE
READ_ERROR
SEVERE_ERROR
SHELF_WRITE_ERROR
WEBSERVICE_IN_USE
WEBSERVICE_NOT_FOUND

RESPONSE
Values for the parameter are:
OK
EXCEPTION
DISASTER

```
INVALID
KERNERROR
PURGED
```

Pipeline Manager domain's generic gates

Table 82 summarizes the Pipeline Manager domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 82. Pipeline Manager domain's generic gates

Gate	Trace	Function	Format
PIDM	PI 0100 PI 0101	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
PIST	PI 0200 PI 0201	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

Modules

Module	Function
DFHPIAP	Remote stub program.
DFHPIAT	Supports PI domain's atomic transactions functions.
DFHPICC	Marshal XML body to COMMAREA and channel data.
DFHPIDM	Domain initialization and termination program.
DFHPIDSH	The pipeline HTTP inbound router module. Starts a service provider pipeline by issuing a DFHPIPM START_PIPELINE call to the pipeline manager.
DFHPIDUF	PI domain dump formatting program.
DFHPIEP	SOAP envelope SAX parser.
DFHPIII	ICM interpreter.
DFHPIIT	PI installation assist transaction program
DFHPIIW	Pipeline manager support for PIIW gate.
DFHPILN	Pipeline callback program
DFHPIPA	SOAP envelope SAX parser
DFHPIPL	PIPL gate functions
DFHPIPM	Pipleine manager domain gate
DFHPIRT	The pipeline HTTP outbound router module. Starts a service requester pipeline by issuing a DFHPIPM START_PIPELINE call to the pipeline manager.
DFHPISB	DFHPICC's callback stub.
DFHPISF	SOAP fault API support.
DFHPISN	SOAP node support.
DFHPISN1	SOAP 1.1 handler program.
DFHPISN2	SOAP 1.2 handler program.
DFHPIST	Pipeline manager's statistics gate.
DFHPITH	The pipeline HTTP transport management program which performs the functions of the PITG gate.
DFHPITL	Top level Web service module

Module	Function
DFHPITP	PI domain's EXEC layer program
DFHPITQ	MQ transport.
DFHPITQ1	CICS SOAP MQ Transport program.
DFHPITRI	PI domain trace formatting program.
DFHPIWR	WEBSERVICE resource functions.
DFHPIWT	Work request manager.
DFHPIXE	SAX event handler interface.

Exits

None

Trace

Chapter 97. Partner resource manager

The partner resource manager (an OCO component of the AP domain) is responsible for managing all operations involving the partner resource table (PRT). A PARTNER definition is required for every remote partner referenced in SAA communications interface calls (see Chapter 43, “SAA Communications and Resource Recovery interfaces,” on page 347). Partner resources are installed either at system initialization or using CEDA INSTALL, and can be discarded using either the CEMT transaction or EXEC CICS commands.

The partner resource manager is implemented as a set of subroutine interfaces.

Functions provided by the partner resource manager

Table 83 summarizes the external subroutine interfaces provided by the partner resource manager. It shows the subroutine call formats, the level-1 trace point IDs of the modules providing the functions for these formats, and the functions provided.

Table 83. Partner resource manager’s subroutine interfaces

Format	Trace	Function
PRCM	AP 0F36 AP 0F37	INQUIRE_PARTNER START_PARTNER_BROWSE GET_NEXT_PARTNER END_PARTNER_BROWSE
PRFS	AP 0F34 AP 0F35	LOCATE_AND_LOCK_PARTNER
PRIN	AP 0F20 AP 0F21	START_INIT COMPLETE_INIT
PRPT	AP 0F30 AP 0F31	ADD_REPLACE_PARTNER DELETE_PARTNER

PRCM format, INQUIRE_PARTNER function

The INQUIRE_PARTNER function of the PRCM format is used to retrieve the installed definition of a specified partner, consisting of the remote transaction program name (TP name), network identifier, netname (network LU name), and profile name.

Input parameters

PARTNER_NAME is the 8-character name of the entry whose contents are to be retrieved.
TP_NAME is a buffer for the output TP name.

Output parameters

NETWORK is the 8-character network identifier.
NETNAME is the 8-character netname.
PROFILE_NAME is the 8-character CICS profile name.
RESPONSE is the subroutine’s response to the call. It can have any of these values:

`OK|EXCEPTION|KERNERROR`

[REASON] is returned when RESPONSE is EXCEPTION. It has this value:

`PARTNER_NOT_FOUND`

PRCM format, START_PARTNER_BROWSE function

The START_PARTNER_BROWSE function of the PRCM format is used to initiate a browse of the partner resource table. The browse starts at the beginning of the table.

Input parameters

None.

Output parameters

BROWSE_TOKEN is the token identifying the browse session initiated by this call.

RESPONSE is the subroutine's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. It has this value:

GETMAIN_FAILED

PRCM format, GET_NEXT_PARTNER function

The GET_NEXT_PARTNER function of the PRCM format is used to retrieve the information stored in the next partner found in a sequential browse of the partner resource table.

Input parameters

BROWSE_TOKEN is the token identifying this browse session.

TP_NAME is a buffer for the output TP name.

Output parameters

PARTNER_NAME is the 8-character name of the entry retrieved.

NETWORK is the 8-character network identifier.

NETNAME is the 8-character netname.

PROFILE_NAME is the 8-character CICS profile name.

RESPONSE is the subroutine's response to the call. It can have any of these values:

OK|EXCEPTION|KERNERROR

[REASON] is returned when RESPONSE is EXCEPTION. It has this value:

END_OF_LIST

PRCM format, END_PARTNER_BROWSE function

The END_PARTNER_BROWSE function of the PRCM format is used to terminate a browse of the partner resource table.

Input parameters

BROWSE_TOKEN is the token identifying this browse session.

Output parameters

RESPONSE is the subroutine's response to the call. It can have either of these values:

OK|KERNERROR

PRFS format, LOCATE_AND_LOCK_PARTNER function

The LOCATE_AND_LOCK_PARTNER function of the PRFS format is used to retrieve the information stored in a named entry in the partner resource table. A table manager read lock is applied to the entry.

Input parameters

PARTNER_NAME is the 8-character name of the entry whose contents are to be retrieved.

TP_NAME is a buffer for the output TP name.

Output parameters

NETWORK is the 8-character network identifier.

NETNAME is the 8-character netname.

PROFILE_NAME is the 8-character CICS profile name.

RESPONSE is the subroutine's response to the call. It can have any of these values:

OK|EXCEPTION|KERNERROR

[REASON] is returned when RESPONSE is EXCEPTION. It has this value:

PARTNER_NOT_FOUND

PRIN format, START_INIT function

The START_INIT function of the PRIN format is used to attach a CICS task to perform initialization of the partner resource manager.

Input parameters

None.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. It can have either of these values:

RESPONSE	Possible REASON values
DISASTER	GETMAIN-FAILED ADD_SUSPEND_FAILED

PRIN format, COMPLETE_INIT function

The COMPLETE_INIT function of the PRIN format is used to wait for the initialization task attached by the START_INIT function to complete processing.

Input parameters

None.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. It has this value:

INIT_TASK_FAILED

PRPT format, ADD_REPLACE_PARTNER function

The ADD_REPLACE_PARTNER function of PRPT format is used to add a named entry to the partner resource table. The new entry replaces the existing entry (if any) with the specified name.

Input parameters

PARTNER_NAME
is the 8-character name of the entry whose contents are to be added or replaced.

NETWORK
is the 8-character network identifier.

NETNAME
is the 8-character netname.

PROFILE_NAME
is the 8-character CICS profile name.

TP_NAME
specifies the address and length of a buffer containing the TP name.

SYSTEM_STATUS
specifies the status of the CICS system at the time of the call. It can have any one of these values (ONLINE here means during execution):

COLD_START|WARM_START|ONLINE

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	CATALOG_WRITE_FAILED GETMAIN_FAILED
EXCEPTION	PARTNER_IN_USE

PRPT format, DELETE_PARTNER function

The DELETE_PARTNER function of the PRPT format is used to delete a named entry in the partner resource table.

Input parameters

PARTNER_NAME
is the 8-character name of the entry to be deleted.

SYSTEM_STATUS
is the status of the CICS system at the time of the call. It can have any one of these values (ONLINE here means during execution):

COLD_START|WARM_START|ONLINE

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	CATALOG_DELETE_FAILED
EXCEPTION	PARTNER_IN_USE PARTNER_NOT_FOUND

Modules

Module	Function
DFHAPTRR	Interprets partner resource manager trace entries
DFHPRCM	Handles the following requests: INQUIRE_PARTNER START_PARTNER_BROWSE GET_NEXT_PARTNER END_PARTNER_BROWSE
DFHPRDUF	Formats the partner resource manager control blocks in a CICS system dump
DFHPRFS	Handles the following request: LOCATE_AND_LOCK_PARTNER
DFHPRIN1	Handles the following requests: START_INIT COMPLETE_INIT
DFHPRIN2	Runs as a CICS task to perform initialization of the partner resource manager
DFHPRPT	Handles the following requests: ADD_REPLACE_PARTNER DELETE_PARTNER
DFHPRRP	Initializes the partner resource table at CICS startup

Exits

No global user exit points are provided for this component.

Trace

The following point ID is provided for the partner resource manager:

- AP 0F20 through AP 0F3F, for which the trace levels are AP 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 98. Partner domain (PT)

The partner domain provides services to coordinate flows between two CICS tasks.

Partner domain's specific gates

Table 84 summarizes the PT domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 84. Partner domain's specific gates

Gate	Trace	Function	XPI
PTTW	OT 0100	CREATE_POOL	NO
	OT 0101	DESTROY_POOL	NO
		QUERY_POOL	NO
		START_POOL_BROWSE	NO
		GET_NEXT_POOL	NO
		END_POOL_BROWSE	NO
		CREATE_PARTNERSHIP	NO
		DESTROY_PARTNERSHIP	NO
		SET_USER_TOKEN	NO
		INQUIRE_USER_TOKEN	NO
		MAKE_PARTNERSHIP	NO
		BREAK_PARTNERSHIP	NO
		TRIGGER_PARTNER	NO
		WAIT_FOR_PARTNER	NO
		QUERY_PARTNERSHIP	NO
		SET_GARBAGE_INTERVAL	NO
		INQUIRE_GARBAGE_INTERVAL	NO

PTTW gate, CREATE_POOL function

The CREATE_POOL function creates a pool for state_tokens.

Input parameters

POOL_NAME
The eight character name of the pool. This name must be unique across all pools. There is no enforced character set for this name.

GARBAGE_COLLECTION
Whether or not garbage collection is to be performed for state_tokens in this pool. It can have either of these two values:

ON|OFF

[GARBAGE_COLLECT_INTERVAL]
The interval in milliseconds between collections of garbage for this pool. If garbage collection is on, this parameter must be provided. If garbage collection is off, this parameter is ignored.

[FREE_USER_DATA_DOMAIN]
An optional callback routine that may be called to free any user data addressed from the user_data_token associated with each state_token. This callback must implement the PTFD FREE_USER_DATA gate.

[FREE_USER_DATA_GATE]
An optional callback routine that may be called to free any user data addressed from the user_data_token associated with each state_token. This callback must implement the PTFD FREE_USER_DATA gate.

Output parameters

POOL_TOKEN
The token of this pool

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NAME_NOT_UNIQUE BAD_CALLBACK

PTTW gate, DESTROY_POOL function

Destroys a pool of state_tokens.

Input parameters

POOL_TOKEN The token of this pool

DESTROY_OPTION
Specifies how the pool is destroyed. It can have any of these values:

MUST_BE_EMPTY|FORCE|QUIESCE

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	POOL_NOT_EMPTY POOL_NOT_FOUND POOL_QUIESCING

PTTW gate, QUERY_POOL function

Query the attributes and state of a pool.

Input parameters

[POOL_NAME] The eight character name of the pool. This name must be unique across all pools. There is no enforced character set for this name.

[POOL_TOKEN] The token of this pool

Output parameters

[POOL_NAME_OUT]
The pool name is returned.

[POOL_TOKEN_OUT]
The pool token is returned.

[POOL_STATE] The cureent state of the pool. It can have any of these values:

EMPTY|NOT_EMPTY|QUIESCING

[GARBAGE_COLLECTION]
Whether or not garbage collection is to be performed for state_tokens in this pool. It can have either of these two values:

ON|OFF

[GARBAGE_COLLECT_INTERVAL]
The interval in milliseconds between collections of garbage for this pool. If garbage collection is on, this parameter must be provided. If garbage collection is off, this parameter is ignored.

[FREE_USER_DATA_DOMAIN]
An optional callback routine that may be called to free any user data addressed from the user_data_token associated with each state_token. This callback must implement the PTFD FREE_USER_DATA gate.

[FREE_USER_DATA_GATE]
An optional callback routine that may be called to free any user data addressed from the user_data_token associated with each state_token. This callback must implement the PTFD FREE_USER_DATA gate.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	POOL_NOT_FOUND

PTTW gate, START_POOL_BROWSE function

Creates a pool cursor to browse pools.

Output parameters

POOL_CURSOR The browse cursor returned from start_pool_browse

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_POOLS

PTTW gate, GET_NEXT_POOL function

Get the next pool

Input parameters

POOL_CURSOR The browse cursor returned from start_pool_browse

Output parameters

[POOL_TOKEN] The token of this pool

[POOL_NAME] The eight character name of the pool. This name must be unique across all pools. There is no enforced character set for this name.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	END_BROWSE INVALID_CURSOR

PTTW gate, END_POOL_BROWSE function

End a browse of pools.

Input parameters

POOL_CURSOR The browse cursor returned from start_pool_browse

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_CURSOR

PTTW gate, CREATE_PARTNERSHIP function

Create a new state block to represent a partnership, and add it to the pool.

Input parameters

POOL_TOKEN The token of this pool

Output parameters

STATE_TOKEN The state_token used to manage the handshake

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	POOL_NOT_FOUND POOL_QUIESCING

PTTW gate, DESTROY_PARTNERSHIP function

Remove a state block from its pool and delete it to destroy the partnership. If the state token is still in use by the partner, it is flagged as deleted.

Input parameters

STATE_TOKEN The state_token used to manage the handshake

Output parameters

[OLD_TRIGSTATE1]
The state of partner 1 before the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[OLD_TRIGSTATE2]
The state of partner 2 before the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[NEW_TRIGSTATE1]
The state of partner 1 after the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[NEW_TRIGSTATE2]
The state of partner 2 after the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND PARTNER_WAITING

PTTW gate, SET_USER_TOKEN function

Change the user token in the state block.

Input parameters

STATE_TOKEN The state_token used to manage the handshake

USER_TOKEN The user token to be associated with the state token

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND

PTTW gate, INQUIRE_USER_TOKEN function

Get the user token in the state block.

Input parameters

STATE_TOKEN The state_token used to manage the handshake

Output parameters

USER_TOKEN The user token to be associated with the state token

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND

PTTW gate, MAKE_PARTNERSHIP function

Establish a partnership with another task. The partner task may or may not have previously made the partnership.

Input parameters

STATE_TOKEN The state_token used to manage the handshake

ORDER Specifies the order in which the partners make the partnership. It can have any of these values:

DONT_CARE|ONLY|SUBSEQUENT

Output parameters

[OLD_TRIGSTATE1]
The state of partner 1 before the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[OLD_TRIGSTATE2]
The state of partner 2 before the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[NEW_TRIGSTATE1]
The state of partner 1 after the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[NEW_TRIGSTATE2]
The state of partner 2 after the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_ONLY NOT_SUBSEQUENT ALREADY_MADE ALREADY_PARTNER NOT_FOUND NOT_PARTNER

PTTW gate, BREAK_PARTNERSHIP function

Break an established partnership.

Input parameters

STATE_TOKEN The state_token used to manage the handshake

[COMPLETION_CODE]
The completion code to be passed to the partner. The caller can use this to notify partner why the partnership is being broken. Once read the completion code is reset to zero. This is optional so that the caller can pass exactly one completion code when calling trigger_partner followed by break_partnership. The completion code is ignored if the resulting state is not_made.

Output parameters

[OLD_TRIGSTATE1]
The state of partner 1 before the request. It can have any of these values:
UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[OLD_TRIGSTATE2]
The state of partner 2 before the request. It can have any of these values:
UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[NEW_TRIGSTATE1]
The state of partner 1 after the request. It can have any of these values:
UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[NEW_TRIGSTATE2]
The state of partner 2 after the request. It can have any of these values:
UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

PARTNER_COMPLETION_CODE
The partner's completion code indicates why the partner broke the partnership.

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_PARTNER NOT_FOUND PARTNERSHIP_NOT_MADE

PTTW gate, TRIGGER_PARTNER function

Notify a waiting partner. If the partner is not actually waiting when trigger is called, the partner will be triggered when it next waits.

Input parameters

STATE_TOKEN The state_token used to manage the handshake

COMPLETION_CODE
The completion code to be passed to the partner.

PARTNER_EXISTENCE
Specifies whether the partner must exist for this request. It can have either of these two values:
DONT_CARE|MUST_EXIST

Output parameters

[OLD_TRIGSTATE1]
The state of partner 1 before the request. It can have any of these values:
UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[OLD_TRIGSTATE2]
The state of partner 2 before the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[NEW_TRIGSTATE1]
The state of partner 1 after the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[NEW_TRIGSTATE2]
The state of partner 2 after the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NAME_NOT_UNIQUE BAD_CALLBACK

PTTW gate, WAIT_FOR_PARTNER function

Wait to be notified by a partner or until the wait times out.

Input parameters

STATE_TOKEN
The state_token used to manage the handshake

PARTNER_EXISTENCE
Specifies whether the partner must exist for this request. It can have either of these two values:

DONT_CARE|MUST_EXIST

[TIMEOUT]
An optional maximum time to wait before waking up in milliseconds

[PURGEABLE]
Specifies whether the wait can be purged. It can have either of these two values:

YES|NO

Output parameters

PARTNER_COMPLETION_CODE
The partner's completion code indicates why the partner broke the partnership.

[OLD_TRIGSTATE1]
The state of partner 1 before the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[OLD_TRIGSTATE2]
The state of partner 2 before the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[NEW_TRIGSTATE1]
The state of partner 1 after the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[NEW_TRIGSTATE2]
The state of partner 2 after the request. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NAME_NOT_UPARTNERSHIP_NOT_MADE NOT_PARTNER TIMED_OUT NOT_FOUND PARTNER_WAITING PARTNER_NOT_THERENIQUE BAD_CALLBACK

PTTW gate, QUERY_PARTNERSHIP function

Get the status of the partner task.

Input parameters

STATE_TOKEN The state_token used to manage the handshake

Output parameters

[XM_TOKEN] The partner's transaction manager token.

[POOL_TOKEN] The token of this pool

[STATE] Describes whether the state token is not made, made or partially made and who by. It can have any of these values:

NOT_MADE|MADE_BY_PARTNER|MADE_BY_SELF|MADE

[STATUS_OF_PARTNER]
Describes whether partner is waiting or has been triggered. It can have any of these values:

UNDEFINED|VALID|TRIGGERED|WAITING|RESUMED

[STATUS_OF_SELF]
Describes whether the caller has been triggered or not. It can have any of these values:

UNDEFINED|VALID|TRIGGERED

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_PARTNER NOT_FOUND

PTTW gate, SET_GARBAGE_INTERVAL function

Set garbage collection interval.

Input parameters

POOL_TOKEN The token of this pool

GARBAGE_COLLECT_INTERVAL
The interval in milliseconds between collections of garbage for this pool. If garbage collection is on, this parameter must be provided. If garbage collection is off, this parameter is ignored.

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	GARBAGE_COLLECTION_OFF POOL_NOT_FOUND

PTTW gate, INQUIRE_GARBAGE_INTERVAL function

Get garbage collection interval.

Input parameters

POOL_TOKEN
The token of this pool

Output parameters

GARBAGE_COLLECTION
Indicates whether or not garbage collection is to be performed for state_tokens in this pool. It can have either of these two values:

ON|OFF

GARBAGE_COLLECT_INTERVAL
The interval in milliseconds between collections of garbage for this pool. If garbage collection is on, this parameter must be provided. If garbage collection is off, this parameter is ignored.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NAME_NOT_UNIQUE BAD_CALLBACK

Modules

Module	Function
DFHPTDM	Domain initialisation and termination. PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHPTTW	Handles the following requests: CREATE_POOL DESTROY_POOL QUERY_POOL START_POOL_BROWSE GET_NEXT_POOL END_POOL_BROWSE CREATE_PARTNERSHIP DESTROY_PARTNERSHIP SET_USER_TOKEN INQUIRE_USER_TOKEN MAKE_PARTNERSHIP BREAK_PARTNERSHIP TRIGGER_PARTNER WAIT_FOR_PARTNER QUERY_PARTNERSHIP SET_GARBAGE_INTERVAL INQUIRE_GARBAGE_INTERVAL

Exits

None

Trace

The point IDs for the PT domain are of the form OTxxxx; the corresponding trace levels are PT 1, PT 2 and Exc.

For more information about the trace points, see *CICS Trace Entries*. For more information about using traces in problem determination, see *CICS Problem Determination Guide*.

Chapter 99. Recovery Manager Domain (RM)

Recovery Manager (RM) is a domain which is responsible for ensuring that the resource updates for a unit of work are all committed or all backed out, including updates across multiple systems.

Resource Owners, such as File Control, are responsible for processing update requests from applications and for backing out updates. Recovery Manager provides interfaces which Resource Owners use to participate in a unit of work. So Recovery Manager coordinates the Resource Owners ensuring that they all either commit or back out the updates for a particular unit of work. Each Resource Owner protects Recovery Manager from the details of how its resources are managed.

Updates on multiple systems are also coordinated by Recovery Manager. However, since systems are connected in a variety of ways, Recovery Manager uses Recovery Manager Connectors (RMCs) to communicate with remote systems. RMCs, such as the LU 6.2 RMC, are responsible for adapting the Recovery Manager protocols to the inter-system protocols. RMCs protect Recovery Manager from the details of the various inter-system protocols.

Additionally, Recovery Manager supports failures such as a system crash, a remote connection failure, or a local resource failure (e.g. an I/O error). It also supports the forward recovery of local resources allowing them to be reconstructed to a consistent state.

Recovery Manager Domain’s specific gates

Table 85 summarizes the Recovery Manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 85. Recovery Manager domain’s specific gate

Gate	Trace	Function	XPI
RMUW	RM 0201	CREATE_UOW	NO
	RM 0202	INQUIRE_UOW_ID	NO
		INQUIRE_UOW_TOKEN	NO
		INQUIRE_UOW	NO
		SET_UOW	NO
		COMMIT_UOW	NO
		FORCE_UOW	NO
		START_UOW_BROWSE	NO
		GET_NEXT_UOW	NO
		END_UOW_BROWSE	NO
		BACKOUT_UOW	NO
		BIND_UOW_TO_TXN	NO
		REATTACH_REPLY	NO
RMLN	RM 0301	ADD_LINK	NO
	RM 0302	DELETE_LINK	NO
		INQUIRE_LINK	NO
		SET_LINK	NO
		ISSUE_PREPARE	NO
		INBOUND_FLOW	NO
		INITIATE_RECOVERY	NO
		SET_RECOVERY_STATUS	NO
		REPORT_RECOVERY_STATUS	NO
		TERMINATE_RECOVERY	NO
		SET_MARK	NO
		START_LINK_BROWSE	NO
		GET_NEXT_LINK	NO
		END_LINK_BROWSE	NO
RMNM	RM 0161	INQUIRE_LOGNAME	NO
	RM 0162	SET_LOGNAME	NO
		CLEAR_PENDING	NO
RMCD	RM 0121	REGISTER	NO
	RM 0122	SET_GATE	NO
		INQUIRE_CLIENT_DATA	NO
		SET_CLIENT_DATA	NO

Table 85. Recovery Manager domain's specific gate (continued)

Gate	Trace	Function	XPI
RMDM	RM 0101 RM 0102	INQUIRE_STARTUP SET_STARTUP SET_LOCAL_LU_NAME SET_PARAMETERS	NO NO NO NO
RMKD	RM 0231 RM 0232	KEYPOINT_DATA	NO
RMRE	RM 0231 RM 0232	APPEND FORCE REMOVE AVAIL REQUEST_FORGET	NO NO NO NO NO
RMSL	RM 06E1 RM 06E2	TAKE_ACTIVITY_KEYPOINT	NO
RMWT	RM 0201 RM 0202	INQUIRE_WORK_TOKEN SET_WORK_TOKEN START_WORK_TOKEN_BROWSE GET_NEXT_WORK_TOKEN END_WORK_TOKEN_BROWSE	NO NO NO NO NO

RMUW gate, CREATE_UOW function

Create a unit of work object under the currently executing transaction.

Input parameters

UOW_ID
An optional parameter specifying the network UOWID to be given to the unit of work object. This parameter will be present if the unit of work being created is part of a distributed unit of work that originated on another system.

HEURISM
An optional parameter specifying whether the unit of work should take a unilateral decision if a failure occurs in the in doubt window? It can have any one of these values:

YES|NO

CHOICE
An optional parameter specifying whether the unit of work should commit or backout if requested to take a unilateral decision. It can have any one of these values:

FORWARD|BACKWARD

INDOUBT_TIMEOUT_INTERVAL
An optional parameter specifying the period of time that the unit of work should be prepared to wait in doubt.

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMUW gate, INQUIRE_UOW_ID function

Return the network and local UOWIDs of the unit of work of the currently executing transaction.

Input parameters

UOW_ID
An optional parameter specifying a buffer in which the network UOWID will be returned.

Output parameters

LOCAL_UOW_ID
An optional parameter to receive the local UOWID.

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMUW gate, INQUIRE_UOW_TOKEN function

Return the token identifying the unit of work object with the specified local UOWID.

Input parameters

LOCAL_UOW_ID The local UOWID of the required unit of work.

Output parameters

UOW_TOKEN A token identifying the unit of work object.

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND

RMUW INQUIRE_UOW function

This function is used to query information about a particular unit of work.

Input parameters

UOW_TOKEN An optional parameter specifying a token used to identify the unit of work object being queried.

TRANSACTION_TOKEN
An optional parameter specifying a token of a transaction whose unit of work object is to be queried.

LOG_CHAIN_TOKEN
An optional parameter specifying a token of a log chain whose unit of work object is to be queried.

UOW_ID An optional parameter specifying a buffer in which the network UOWID will be returned.

LOGNAME An optional parameter specifying a buffer in which the log name of the coordinating system will be returned.

LOCAL_ACCESS_ID
An optional parameter specifying a buffer in which the local access id of resource causing the unit of work to shunt will be returned.

REMOTE_ACCESS_ID
An optional parameter specifying a buffer in which the netname of coordinating system will be returned.

LINK_ID An optional parameter specifying a buffer in which the termid of the link to the coordinating system will be returned.

Output parameters

OUT_UOW_TOKEN
The token used to identify the unit of work object.

LOCAL_UOW_ID The local unit of work id.

TRANID The tranid of the task that created the unit of work object.

TERMID The termid associated with the task that created the unit of work object.

TERMINAL_LUNAME
The terminal LU name associated with the task that created the unit of work object.

USERID The userid associated with the task that created the unit of work object.

CHOICE The choice of whether the unit of work should commit or backout if requested to take a unilateral decision. It can have any one of these values:

FORWARD|BACKWARD

UOW_STATUS The status of the unit of work. It can have any one of these values:

FORWARD|BACKWARD|IN_DOUBT|IN_FLIGHT|
HEURISTIC_FORWARD|HEURISTIC_BACKWARD

SHUNTED The unit of work may or may not be shunted. It can have any one of these values:

YES|NO

DURATION An 8 byte Store Clock representation of the time the unit of work changed state.

CREATION_TIME
An 8 byte Store Clock representation of the time the unit of work was created.

CLIENT_NAME
The name of the Recovery Manager client that owns the resource that has caused the unit of work to shunt.

ACCESS_ID_TYPE
The type of resource that has caused the unit of work to shunt. It can have any one of these values:

LOCAL|REMOTE

TRANNUM
The task number of the task that created the unit of work.

OP_ID
The Operator Id associated with the task that created the unit of work.

FIRST_UOW_FOR_TXN
It can have any one of these values:

YES|NO

HEURISM
Whether the unit of work should take a unilateral decision if a failure occurs in the in doubt window? It can have any one of these values:

YES|NO

AWAITING_FORGET
The unit of work might have completed syncpoint processing, and be merely waiting for confirmation that subordinates have completed theirs. It can have any one of these values:

YES|NO

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND

RMUW gate, SET_UOW function

This function is used to set characteristics of the currently executing unit of work.

Input parameters

HEURISM
Determines whether the unit of work will take a unilateral decision if a failure occurs in the in doubt window, or waits for communication with the coordinating system to be reestablished. It can have any one of these values:

YES|NO

HEURISTIC_CAUSE
An indication of the reason a unilateral decision must be taken. It can have any one of these values:

TD_CLIENT|LU61_CLIENT|MRO_CLIENT|
RMI_CLIENT|OTHER_CLIENT

Output parameters

USERID
When requested this parameter causes the userid associated with unit of work to be reset to that of the currently executing transaction.

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND

RMUW gate, COMMIT_UOW function

This function attempts to commit the changes made in a unit of work.

Input parameters

CONTINUE
Is the task continuing into a following, new unit of work. This parameter can have any one of these values:

YES|NO

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ROLLBACK, LOCAL_NO_VOTE, REMOTE_NO_VOTE, REMOTE_NO_DECISION, HEURISTIC_READONLY_COMMIT, HEURISTIC_READONLY_BACKOUT, HEURISTIC_BACKOUT, LINKS_INVALID, HEURISTIC_COMMIT, INDOUBT_FAILURE, COMMIT_FAILURE, REMOTE_COMMIT_ABENDED

RMUW gate, FORCE_UOW function

This function forces an in doubt unit of work to unilaterally commit or backout its changes rather than continue waiting for resynchronization with the coordinating system.

Input parameters

UOW_TOKEN
The token identifying the unit of work object.

DIRECTION
Parameter specifying whether to commit (FORWARD), backout (BACKWARD) or obey the ACTION attribute in the definition of the originating transaction. It can have any one of these values:

FORWARD|BACKWARD|HEURISTIC

HEURISTIC_CAUSE
The reason for the force. It can have any one of these values:

OPERATOR|TIMEOUT|OTHER_CAUSE

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND, RESYNCH_IN_PROGRESS, UOW_NOT_INDOUBT

RMUW gate, START_UOW_BROWSE function

This function is used to start a browse of unit of work objects in the system.

Input parameters

SHUNTED
The browse can be of only shunted units of work, only non-shunted units of work or all units of work. This parameter can have any one of these values:

YES|NO|BOTH

Output parameters

BROWSE_TOKEN
A token to be used on subsequent GET_NEXT_UOW calls.

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND

RMUW gate, GET_NEXT_UOW function

This function returns information about the next unit of work object in the browse.

Input parameters

BROWSE_TOKEN
A token obtained from a previous START_UOW_BROWSE call.

UOW_ID
An optional parameter specifying a buffer in which the network UOWID will be returned.

LOGNAME
An optional parameter specifying a buffer in which the log name of the coordinating system will be returned.

LOCAL_ACCESS_ID
An optional parameter specifying a buffer in which the local access id of resource causing the unit of work to shunt will be returned.

REMOTE_ACCESS_ID
An optional parameter specifying a buffer in which the netname of coordinating system will be returned.

LINK_ID
An optional parameter specifying a buffer in which the termid of the link to the coordinating system will be returned.

Output parameters

OUT_UOW_TOKEN
The token used to identify the unit of work object.

LOCAL_UOW_ID
The local unit of work id.

TRANID
The tranid of the task that created the unit of work object.

TERMID
The termid associated with the task that created the unit of work object.

TERMINAL_LUNAME
The terminal LU name associated with the task that created the unit of work object.

USERID
The userid associated with the task that created the unit of work object.

CHOICE
The choice of whether the unit of work should commit or backout if requested to take a unilateral decision. It can have any one of these values:

FORWARD|BACKWARD

UOW_STATUS
The status of the unit of work. It can have any one of these values:

FORWARD|BACKWARD|IN_DOUBT|IN_FLIGHT|
HEURISTIC_FORWARD|HEURISTIC_BACKWARD

SHUNTED
The unit of work may or may not be shunted. It can have any one of these values:

YES|NO

DURATION
An 8 byte Store Clock representation of the time the unit of work changed state.

CREATION_TIME
An 8 byte Store Clock representation of the time the unit of work was created.

CLIENT_NAME The name of the Recovery Manager client that owns the resource that has caused the unit of work to shunt.

ACCESS_ID_TYPE
The type of resource that has caused the unit of work to shunt. It can have any one of these values:

LOCAL|REMOTE

TRANNUM The task number of the task that created the unit of work.

OP_ID The Operator Id associated with the task that created the unit of work.

FIRST_UOW_FOR_TXN
It can have any one of these values:

YES|NO

HEURISM Whether the unit of work should take a unilateral decision if a failure occurs in the in doubt window? It can have any one of these values:

YES|NO

AWAITING_FORGET
The unit of work might have completed syncpoint processing, and be merely waiting for confirmation that subordinates have completed theirs. It can have any one of these values:

YES|NO

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN, BROWSE_END

RMUW gate, END_UOW_BROWSE function

This function is used at the end of a browse of the unit of work objects in the system.

Input parameters

BROWSE_TOKEN A token obtained from a previous START_UOW_BROWSE call.

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN

RMUW gate, BACKOUT_UOW function

This function causes the changes in a unit of work to be backed out.

Input parameters

CONTINUE This parameter indicates whether the task is continuing into a following, new unit of work. This parameter can have any one of these values:

YES|NO

RESTART This parameter is only applicable when CONTINUE(NO) is specified and indicates whether or not transaction restart will be performed.

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BACKOUT_FAILURE, COMMIT_FAILURE, ROLLBACK_NOT_SUPPORTED, REMOTE_COMMIT_ABENDED

RMUW gate, BIND_UOW_TO_TXN function

Make the specified unit of work the current unit of work for the current transaction.

Input parameters

UOW_TOKEN
The token identifying the unit of work object.

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMUW gate, REATTACH_REPLY function

This function gives control to Recovery Manager to do its unshunt processing under a re-attached transaction.

Input parameters

UOW_TOKEN
The token identifying the unit of work object.

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMLN gate, ADD_LINK function

This function adds a link to a remote system to a unit of work. The unit of work is distributed across more than one system and Recovery Manager will manage the syncpoint processing between systems.

Input parameters

CLIENT_NAME
Name of the communications protocol used on the link. It can have any one of these values:

IRC |IRCO|LU61|LU62|RMI |IND

LOGNAME_BUFFER
An optional parameter specifying a buffer containing the logname of the remote system.

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of the remote system, or the name of the External Resource Manager.

LINK_ID_BUFFER
A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier.

LINK_ID_SOURCE
An optional parameter specifying whether the local or remote system allocated the session. It can have any one of these values:

LOCAL|REMOTE

RMC_TOKEN
A token to be passed to the client on all callback functions.

LAST
A parameter specifying whether the remote system supports the last agent optimization. It can have any one of these values:

YES|NO|MAYBE|DESIRABLE

PRESUMPTION
A parameter specifying whether the remote system assumes the presume abort or presume nothing protocols. It can have any one of these values:

ABORT|NOTHING

PRELOGGING
A parameter specifying whether the client requires to be called with the PERFORM_PRELOGGING callback function. It can have any one of these values:

YES|NO

SINGLE_UPDATER
A parameter specifying whether the remote system supports the single updater optimization. It can have any one of these values:

YES|NO

COORDINATOR
A parameter specifying whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:

YES|NO

INITIATOR
A parameter specifying whether the remote system is the initiator of the syncpoint. It can have any one of these values:

YES|NO

RECOVERY_STATUS
A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system. It can have any one of these values:

NECESSARY|UNNECESSARY|SYNC_LEVEL_1

Output parameters

LINK_TOKEN
A token identifying the new Recovery Manager Link object.

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CLIENT_UNKNOWN, COORDINATOR_ALREADY

RMLN gate, DELETE_LINK function

This function removes a link to a remote system from a unit of work. The remote system will not now be included in syncpoint processing for the current unit of work.

Input parameters

LINK_TOKEN
A token identifying the Recovery Manager Link object.

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LINK_UNKNOWN

RMLN gate, INQUIRE_LINK function

This function returns information about a given Recovery Manager Link object.

Input parameters

LINK_TOKEN
A token identifying a Recovery Manager Link object.

RESOLVE_TO_CURRENT_LINK
Up to two Recovery Manager Link objects may be associated with a token. This optional parameter specifies whether to return information about the most recent or not. It can have any one of these values:

YES|NO

REMOTE_ACCESS_ID_BUFFER
A buffer in which the netname of the remote system, or External Resource Manager name will be returned.

LOGNAME_BUFFER
A buffer in which the logname of the remote system will be returned.

LINK_ID_BUFFER
A buffer in which the termid of the session to the remote system, or External Resource Manager qualifier will be returned.

Output parameters

CLIENT_NAME
The name of the protocol that owns the Recovery Manager Link object. It can have any one of these values:

IRC |IRCO|LU61|LU62|RMI |IND

COORDINATOR
Whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:

YES|NO

INITIATOR
Whether the remote system is the initiator of the syncpoint of the distributed unit of work. It can have any one of these values:

YES|NO

LAST
Whether the remote system supports the last agent optimization. It can have any one of these values:

YES|NO|MAYBE

SINGLE_UPDATER
Whether the remote system supports the single updater optimization. It can have any one of these values:

YES|NO

PRESUMPTION
Whether the remote system assumes the presume abort or presume nothing protocols. It can have any one of these values:

ABORT|NOTHING

RECOVERY_STATUS
Whether recoverable work has taken place as part of the distributed unit of work on the remote system. It can have any one of these values:

NECESSARY|UNNECESSARY|SYNC_LEVEL_1

FORGET
Whether all obligations to the remote system with respect to recovery have been discharged. It can have any one of these values:

YES|NO

MARK
Whether the Recovery Manager Link object has been marked during resynchronization. It can have any one of these values:

YES|NO

UNSHUNTED
Whether the unit of work is not currently shunted. It can have any one of these values:

YES|NO

RESYNC_SCHEDULED
Whether resynchronization activity has been scheduled. It can have any one of these values:

YES|NO

ACCESSIBLE Whether the communications link to the remote system is active or not. It can have any one of these values:

YES|NO|SHUNTED

LINK_ID_SOURCE
Whether the local or remote system allocated the session. It can have any one of these values:

LOCAL|REMOTE

UOW_TOKEN The token identifying the unit of work object.

LOCAL_UOW_ID The local unit of work id of the unit of work to which the Recovery Manager Link object belongs.

HEURISM Whether the unit of work to which the Recovery Manager Link object belongs will take a unilateral decision if a failure occurs in the in doubt window. It can have any one of these values:

YES|NO

RMC_TOKEN A token to be passed to the client on all callback functions.

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LINK_UNKNOWN

RMLN gate, SET_LINK function

This function is used to set characteristics of a Recovery Manager Link object.

Input parameters

LINK_TOKEN A token used to identify a Recovery Manager Link object.

RESOLVE_TO_CURRENT_LINK
Up to two Recovery Manager Link objects may be associated with a token. This optional parameter specifies whether to set characteristics of the most recent or not. It can have any one of these values:

YES|NO

LOGNAME_BUFFER
An optional parameter specifying a buffer containing a logname to be associated with the Recovery Manager Link object.

COORDINATOR A parameter specifying whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:

YES|NO

INITIATOR A parameter specifying whether the remote system is the initiator of the syncpoint. It can have any one of these values:

YES|NO

RECOVERY_STATUS
A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system. It can have any one of these values:

NECESSARY|UNNECESSARY|SYNC_LEVEL_1

SINGLE_UPDATER
A parameter specifying whether the remote system supports the single updater optimization. It can have any one of these values:

YES|NO

PRELOGGING A parameter specifying whether the client requires to be called with the PERFORM_PRELOGGING callback function. It can have any one of these values:

YES|NO

LINK_ID_BUFFER
A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier.

LINK_ID_SOURCE
An optional parameter specifying whether the local or remote system allocated the session. It can have any one of these values:

LOCAL|REMOTE

UNSHUNTED
A parameter specifying whether the unit of work is not currently shunted. It can have any one of these values:

YES|NO

RESYNC_SCHEDULED
A parameter specifying whether resynchronization activity has been scheduled. It can have any one of these values:

YES|NO

ACCESSIBLE
A parameter specifying that the communications link to the remote system has failed. It can have any one of these values:

NO|SHUNTED

FORGET
A parameter specifying whether all obligations to the remote system with respect to recovery have been discharged. It can have any one of these values:

YES|NO

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LINK_UNKNOWN, COORDINATOR_ALREADY, INITIATOR_ALREADY

RMLN gate, ISSUE_PREPARE function

This function performs phase 1 of syncpoint processing on the specified Recovery Manager Link object.

Input parameters

LINK_TOKEN
A token used to identify a Recovery Manager Link object.

CONTINUE
Is the task continuing into a following, new unit of work. This parameter can have any one of these values:

YES|NO

Output parameters

VOTE
The vote from the client owning the Recovery Manager Link object. This parameter can have any one of these values:

YES|NO|NO_CONTINUE|READ_ONLY

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LINK_UNKNOWN, COORDINATOR_ALREADY, INITIATOR_ALREADY, PREPARE_REJECTED

RMLN gate, INBOUND_FLOW function

This function is used to notify Recovery Manager of the successful completion of syncpoint processing on the remote system, or a communications failure with the remote system.

Input parameters

LINK_TOKEN A token used to identify a Recovery Manager Link object.

FLOW A parameter specifying successful completion (DATA) or communication failure (UNBIND). It can have any one of these values:

DATA|UNBIND

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LINK_UNKNOWN, LINK_INACCESSIBLE

RMLN gate, INITIATE_RECOVERY function

This function identifies a Recovery Manager Link object in an in doubt failed unit of work and marks it as being resynchronized.

Input parameters

UOW_ID An optional parameter specifying a buffer containing the network UOWID of the unit of work to be resynchronized.

LOCAL_UOW_ID An optional parameter specifying the local UOWID.

CLIENT_NAME The name of the Recovery Manager client that owns the Recovery Manager Link object over which resynchronization is to take place.

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of the remote system, or the name of the External Resource Manager of the Recovery Manager Link object over which resynchronization is to take place.

LINK_ID_BUFFER
A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier of the Recovery Manager Link object over which resynchronization is to take place.

LINK_ID_SOURCE
An optional parameter specifying whether the local or remote system allocated the session associated with the Recovery Manager Link object over which resynchronization is to take place. It can have any one of these values:

LOCAL|REMOTE

DIRECTION A parameter specifying whether the resynchronization activity was initiated by the local or remote system. It can have any one of these values:

INBOUND|OUTBOUND

Output parameters

UOW_TOKEN
The token identifying the unit of work object. to which the Recovery Manager Link object being resynchronized belongs.

LINK_TOKEN
A token identifying the Recovery Manager Link object being resynchronized.

COORDINATOR
A parameter specifying whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:

YES|NO

INITIATOR
A parameter specifying whether the remote system is the initiator of the syncpoint. It can have any one of these values:

YES|NO

PRESUMPTION
Whether the remote system assumes the presume abort or presume nothing protocols. It can have any one of these values:

ABORT|NOTHING

UOW_STATUS
The status of the unit of work object that the Recovery Manager Link object belongs to. It can have any one of these values:

INDOUBT|FORWARD|BACKWARD|
HEURISTIC_FORWARD|HEURISTIC_BACKWARD

FAILURE_TIME
An 8 byte Store Clock representation of the in doubt failure time.

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LINK_UNKNOWN, RECOVERY_ALREADY_IN_PROG, LINK_ACTIVE

RMLN gate, SET_RECOVERY_STATUS function

This function is used to notify an Recovery Manager Link object of the outcome of a distributed unit of work which failed in the in doubt window. It results in the shunted unit of work the Recovery Manager Link object belongs to unshunting and committing or backing out its resource updates as appropriate.

Input parameters

LINK_TOKEN
A token identifying the Recovery Manager Link object being resynchronized.

DIRECTION
A parameter specifying whether the resynchronization activity was initiated by the local or remote system. It can have any one of these values:

INBOUND|OUTBOUND

REMOTE_UOW_STATUS
The status of the unit of work in the remote system. It can have any one of these values:

INDOUBT|HEURISTIC_FORWARD|HEURISTIC_BACKWARD|
FORWARD|BACKWARD|HEURISTIC_MIXED|COLD|RESET|UNKNOWN

TOLERATE_VIOLATIONS
A parameter specifying the rules to be used to detect resynchronization protocol violations. It can have any one of these values:

YES|NO

Output parameters

UOW_STATUS
The status (as a result of the resynchronization) of the unit of work object to which the Recovery Manager Link object belongs. It can have any one of these values:

INDOUBT|HEURISTIC_FORWARD|HEURISTIC_BACKWARD|
FORWARD|BACKWARD

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LINK_UNKNOWN, RECOVERY_NOT_IN_PROGRESS, ALREADY_SET

RMLN gate, REPORT_RECOVERY_STATUS function

This function is similar to SET_RECOVERY_STATUS but is applicable in the case of Presumed Abort or Last Agent resynchronization where the coordinator has backed out and has no record of the UOW. The participant may have gone indoubt, and needs to resynchronize.

Input parameters

UOW_ID A parameter specifying a buffer containing the network UOWID of the unit of work to be resynchronized.

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of the remote system, or the name of the External Resource Manager of the Recovery Manager Link object over which resynchronization is to take place.

REMOTE_UOW_STATUS
The status of the unit of work in the remote system. It can have any one of these values:

INDOUBT|HEURISTIC_FORWARD|HEURISTIC_BACKWARD|
HEURISTIC_MIXED

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMLN gate, TERMINATE_RECOVERY function

Input parameters

LINK_TOKEN A token identifying the Recovery Manager Link object being resynchronized.

DIRECTION A parameter specifying whether the resynchronization activity was initiated by the local or remote system. It can have any one of these values:

INBOUND|OUTBOUND

FORGET A parameter specifying whether all obligations to the remote system with respect to recovery have been discharged. It can have any one of these values:

YES|NO

OPERATOR_INITIATED
A parameter specifying whether the function is the result of an explicit user action. It can have any one of these values:

YES|NO

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LINK_UNKNOWN, RECOVERY_NOT_IN_PROGRESS, SET_NOT_DONE

RMLN gate, SET_MARK function

This function marks a Recovery Manager Link object during recovery.

Input parameters

LINK_TOKEN A token identifying the Recovery Manager Link object to be marked.

MARK It can have any one of these values:

YES|NO

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LINK_UNKNOWN, LINK_ACTIVE, RECOVERY_IN_PROGRESS

RMLN gate, START_LINK_BROWSE function

This function starts a browse of Recovery Manager Link objects. The browse can return either

- all the Recovery Manager Link objects in the system owned by a particular Recovery Manager client and associated with a particular remote system or External Resource Manager, or
- all Recovery Manager Link objects belonging to a particular unit of work object.

Input parameters

CLIENT_NAME The name of a Recovery Manager client.

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of the remote system, or the name of the External Resource Manager.

UOW_TOKEN The token identifying a unit of work object.

Output parameters

LINK_BROWSE_TOKEN
A token to be used during a browse of all Recovery Manager Link objects for a particular Recovery Manager client.

UOW_BROWSE_TOKEN
A token to be used during a browse of all Recovery Manager Link objects for a particular unit of work object.

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UOW_UNKNOWN, CLIENT_UNKNOWN

RMLN gate, GET_NEXT_LINK function

This function returns information about the next Recovery Manager Link object in a browse.

Input parameters

LINK_BROWSE_TOKEN
A token identifying a browse of all the Recovery Manager Link objects belonging to a particular Recovery Manager client.

UOW_BROWSE_TOKEN
A token identifying a browse of all the Recovery Manager Link objects belonging to a particular unit of work object.

REMOTE_ACCESS_ID_BUFFER
A buffer in which the netname of the remote system, or External Resource Manager name will be returned.

LOGNAME_BUFFER
A buffer in which the logname of the remote system will be returned.

LINK_ID_BUFFER
A buffer in which the termid of the session to the remote system, or External Resource Manager qualifier will be returned.

Output parameters

LINK_TOKEN

CLIENT_NAME
The name of the protocol that owns the Recovery Manager Link object. It can have any one of these values:

IRC |IRCO|LU61|LU62|RMI |IND

COORDINATOR
Whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:

YES|NO

INITIATOR
Whether the remote system is the initiator of the syncpoint of the distributed unit of work. It can have any one of these values:

YES|NO

LAST
Whether the remote system supports the last agent optimization. It can have any one of these values:

YES|NO|MAYBE

SINGLE_UPDATER
Whether the remote system supports the single updater optimization. It can have any one of these values:

YES|NO

PRESUMPTION
Whether the remote system assumes the presume abort or presume nothing protocols. It can have any one of these values:

ABORT|NOTHING

RECOVERY_STATUS
Whether recoverable work has taken place as part of the distributed unit of work on the remote system. It can have any one of these values:

NECESSARY|UNNECESSARY|SYNC_LEVEL_1

FORGET
Whether all obligations to the remote system with respect to recovery have been discharged. It can have any one of these values:

YES|NO

MARK
Whether the Recovery Manager Link object has been marked during resynchronization. It can have any one of these values:

YES|NO

UNSHUNTED
Whether the unit of work is not currently shunted. It can have any one of these values:

YES|NO

RESYNC_SCHEDULED
Whether resynchronization activity has been scheduled. It can have any one of these values:

YES|NO

ACCESSIBLE
Whether the communications link to the remote system is active or not. It can have any one of these values:

YES|NO|SHUNTED

LINK_ID_SOURCE
Whether the local or remote system allocated the session. It can have any one of these values:

LOCAL|REMOTE

UOW_TOKEN
The token identifying the unit of work object.

LOCAL_UOW_ID
The local unit of work id of the unit of work to which the Recovery Manager Link object belongs.

HEURISM
Whether the unit of work to which the Recovery Manager Link object belongs will take a unilateral decision if a failure occurs in the in doubt window. It can have any one of these values:

YES|NO

RMC_TOKEN
A token to be passed to the client on all callback functions.

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UOW_UNKNOWN, END_BROWSE
INVALID	INVALID_BROWSE

RMLN gate, END_LINK_BROWSE function

This function is used to terminate a browse of Recovery Manager Link objects.

Input parameters

LINK_BROWSE_TOKEN
A token identifying a browse of all the Recovery Manager Link objects belonging to a particular Recovery Manager client.

UOW_BROWSE_TOKEN
A token identifying a browse of all the Recovery Manager Link objects belonging to a particular unit of work object.

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_BROWSE

RMNM gate, INQUIRE_LOGNAME function

This function returns the logname and data associated with the specified remote system being communicated with via the specified Recovery Manager client.

Input parameters

CLIENT_NAME Name of a Recovery Manager client.

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of the remote system.

LOGNAME_BUFFER
A buffer to be used to return the logname.

RMC_DATA_BUFFER
A buffer to be used to return data owned by the Recovery Manager client.

Output parameters

IN_USE Whether there are any Recovery Manager Link object in the system associated with the logname. It can have any one of these values:

YES|NO

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND, UNKNOWN_CLIENT

RMNM gate, SET_LOGNAME function

This function is used to associate a logname and some data with the netname of a remote system for a specified Recovery Manager client.

Input parameters

CLIENT_NAME A name of a Recovery Manager client.

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of a remote system.

LOGNAME_BUFFER
A buffer containing the logname to be associated with the netname.

RMC_DATA_BUFFER
A buffer containing data to be associated with the netname.

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_CLIENT

RMNM gate, CLEAR_PENDING function

This function is used to remove Recovery Manager Link objects associated with a specified remote system. Affected indoubt units of work will take a unilateral decision to commit or backout their resource updates.

Input parameters

CLIENT_NAME A name of a Recovery Manager client.

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of the remote system.

COLD A parameter specifying whether the remote system has a new log and so has lost recovery information with respect to units of work in this system. It can have any one of these values:

YES|NO

ALL A parameter specifying whether only Recovery Manager Link objects with the same logname as that currently associated with the remote system should be removed or all Recovery Manager Link objects.

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND

RMCD gate, REGISTER function

This function is used to register a Recovery Manager client.

Input parameters

CLIENT_NAME A name of a Recovery Manager client.

CLIENT_TYPE Whether the client owns local (RO) or remote (RMC) resources. It can have any one of these values:

RO|RMC

GATE An optional parameter specifying the kernel gate that services the client's callback functions.

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ALREADY_REGISTERED, TOO_LATE

RMCD gate, SET_GATE function

This function is used to inform Recovery Manager of the kernel gate that services a Recovery Manager clients callback functions.

Input parameters

CLIENT_NAME A name of a Recovery Manager client.

GATE A parameter specifying the kernel gate that services the client's callback functions.

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_CLIENT, GET_ALREADY_SET

RMCD gate, INQUIRE_CLIENT_DATA function

This function returns data associated with a Recovery Manager client.

Input parameters

CLIENT_NAME A name of a Recovery Manager client.

CLIENT_DATA_BUFFER
A buffer to contain the data returned.

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_CLIENT, CLIENT_DATA_TOO_LONG

RMCD gate, SET_CLIENT_DATA function

This function associates some data with a Recovery Manager client.

Input parameters

CLIENT_NAME A name of a Recovery Manager client.

CLIENT_DATA_BUFFER
A buffer containing the data to be associated with the Recovery Manager client.

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_CLIENT, CLIENT_DATA_TOO_LONG

RMDM gate, INQUIRE_STARTUP function

This function returns information about the type of system start being performed.

Input parameters

None

Output parameters

STARTUP It can have any one of these values:

COLD|WARM|EMERGENCY

ALL A value specifying whether all components are cold starting. It can have any one of these values:

YES|NO

INITIAL_START
A value specifying whether the cold start is in fact an initial one. It can have any one of these values:

YES|NO

LAST_COLD_START_TIME
An 8 byte Store Clock representation of the last cold start time.

LAST_EMER_START_TIME
An 8 byte Store Clock representation of the last emergency start time.

LAST_INIT_START_TIME
An 8 byte Store Clock representation of the last initial start time.

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMDM gate, SET_STARTUP function

This function sets the type of start that will be performed when this system is next restarted.

Input parameters

STARTUP
The type of start. It can have any one of these values:

COLD|NORESTART

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMDM gate, SET_LOCAL_LU_NAME function

This function sets the local LU name, that is used in the generation of network UOWIDs by in this system.

Input parameters

LOCAL_LU_NAME
A parameter specifying the local LU name.

LOCAL_LU_NAME_LENGTH
A parameter specifying the length of the local LU name.

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMDM gate, SET_PARAMETERS function

This function is used only by Parameter Manager Domain to inform Recovery Manager of initialization parameters.

Input parameters

DELETE_LOG
An optional parameter specifying whether an initial start has been requested in the System Initialization Table, and so the contents of the system log should be deleted. It can have any one of these values:

YES|NO

STARTUP
An optional parameter used in the case where OFFSITE=YES has been specified as a SIT override. It can only have the value EMERGENCY.

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMKD gate, KEYPOINT_DATA function

This function writes Recovery Manager client data to the system log for keypointing purposes.

Input parameters

CLIENT_NAME
A name of a Recovery Manager client.

DATA
Address of an extended Iliffe vector. An extended Iliffe vector consists of a linked list of at least one element. Each element of the linked list consists of a variable length array of address length pairs. Each address and length field is four bytes long. The top bit of each address is off except for the last which may be on.

If an address is binary zero, then this terminates the element and the linked list.

If an address has the top bit on, then it terminates the element and points to the next element in the linked list.

An extended Iliffe vector simply represents the block of data formed by concatenating all the blocks which are pointed to by address length pairs in the vector which have the address top bit off. The order is from front to back of the linked list and from low to high index within each array.

REMARK
An optional parameter for the benefit of trace to describe the data being logged.

RAISE_INV_DATA_LENGTH
An optional parameter specifying whether the caller wishes to be informed of there being to much data to be logged. It can have any one of these values:

YES|NO

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_DATA_LENGTH, INVALID_CLIENT_NAME, NO_DATA

RMRE gate, APPEND function

This function writes data to the system log. The data written is associated with the current unit of work of the currently executing transaction if either FORWARD_DATA(YES) or BACKWARD_DATA(YES) is specified.

Input parameters

CLIENT_NAME
A name of a Recovery Manager client.

RESOURCE_ID
A parameter specifying the name of the resource with which the data to be logged is associated.

DATA
Address of an extended Iliffe vector. An extended Iliffe vector consists of a linked list of at least one element. Each element of the linked list consists of a variable length array of address length pairs. Each address and length field is four bytes long. The top bit of each address is off except for the last which may be on.

If an address is binary zero, then this terminates the element and the linked list.

If an address has the top bit on, then it terminates the element and points to the next element in the linked list.

An extended Iliffe vector simply represents the block of data formed by concatenating all the blocks which are pointed to by address length pairs in the vector which have the address top bit off. The order is from front to back of the linked list and from low to high index within each array.

FORCE_DATA
A parameter specifying whether the data is forced out on to the non-volatile log or can merely be written to the volatile log buffer. It can have any one of these values:

YES|NO

FORWARD_DATA
A parameter specifying whether the data is used for forward recovery purposes. It can have any one of these values:

YES|NO

BACKWARD_DATA
A parameter specifying whether the data is used for backward recovery purposes. It can have any one of these values:

YES|NO

REMARK
An optional parameter for the benefit of trace to describe the data being logged.

LOG_BUFFER_SUSPEND
An optional parameter specifying whether the caller can tolerate the task suspending to wait for space in a log buffer. It can have any one of these values:

YES|NO

RAISE_INV_DATA_LENGTH
An optional parameter specifying whether the caller wishes to be informed of there being to much data to be logged. It can have any one of these values:

YES|NO

Output parameters

FORCE_TOKEN
A token that can be used to force the data on to the non-volatile log with the FORCE function of the RMRE gate.

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_DATA_LENGTH, INSUFFICIENT_BUFFER_SPACE, INVALID_CLIENT_NAME, INVALID_RESOURCE_ID, NO_DATA

RMRE gate, FORCE function

This function forces data written previously to a log buffer to the non-volatile log.

Input parameters

FORCE_TOKEN
A token returned on a previous call to the APPEND function of the RMRE gate.

Output parameters

RESPONSE
is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMRE gate, REMOVE function

This function removes data logged by a Recovery Manager client and associated with a particular local resource from a unit of work.

Input parameters

UOW_ID The network UOWID under which the data was logged.

LOCAL_UOW_ID The local UOWID under which the data was logged.

CLIENT_NAME The name of the Recovery Manager client that logged the data.

LOCAL_ACCESS_ID
The name of the local resource with which the logged data was associated.

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UOW_NOT_SHUNTED, UOW_NOT_BACKWARDS, INVALID_CLIENT_NAME, INVALID_LOCAL_ACCESS_ID

RMRE gate, AVAIL function

This function informs Recovery Manager that a local resource has become available. It is used when either a backout failure or a commit failure has previously occurred and the resource (or reason for the failure) has now cleared - or there is now reason to believe it may have cleared.

Input parameters

CLIENT_NAME The name of the Recovery Manager client that owns the local resource.

LOCAL_ACCESS_ID
The name of the local resource.

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LOCAL_ACCESS_ID_UNKNOWN

RMRE gate, REQUEST_FORGET function

This function associates a Recovery Manager client and a named local resource with a requirement to engage in forget processing.

Input parameters

CLIENT_NAME The name of the Recovery Manager client that owns the local resource.

LOCAL_ACCESS_ID
The name of the local resource.

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_CLIENT_NAME, INVALID_LOCAL_ACCESS_ID

RMSL gate, TAKE_ACTIVITY_KEYPOINT function

This function performs the activity associated with taking a keypoint.

Input parameters

None

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	None

RMWT gate, INQUIRE_WORK_TOKEN function

This function returns the value of the work token belonging to the named Recovery Manager client in a particular unit of work object.

Input parameters

UOW_TOKEN An optional parameter specifying the token identifying a unit of work object. If not specified the work token from the current unit of work of the currently executing transaction is returned.

CLIENT_NAME The name of a Recovery Manager client.

Output parameters

WORK_TOKEN The value of the Recovery Manager clients work token in the specified unit of work object.

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND

RMWT gate, START_WORK_TOKEN_BROWSE function

This function starts a browse of a all the non-zero work tokens in the system for a specific Recovery Manager client.

Input parameters

CLIENT_NAME The name of a Recovery Manager client.

Output parameters

BROWSE_TOKEN A token to be used during the browse.

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND

RMWT gate, GET_NEXT_WORK_TOKEN function

This function returns the next non-zero work token for the Recovery Manager client specified on the START_WORK_TOKEN_BROWSE. The token used to identify the unit of work object and local UOWID associated with the work token are also optionally returned.

Input parameters

BROWSE_TOKEN A token identifying the browse.

Output parameters

WORK_TOKEN The value of the Recovery Manager clients work token.

UOW_TOKEN The token identifying the unit of work object.

LOCAL_UOW_ID The local UOWID.

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN, BROWSE_END

RMWT gate, END_WORK_TOKEN_BROWSE function

This function terminates a browse of work tokens.

Input parameters

BROWSE_TOKEN A token identifying the browse.

Output parameters

RESPONSE is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_BROWSE_TOKEN

Recovery Manager domain's generic gates

Table 86 summarizes the Recovery Manager domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 86. Recovery Manager domain's generic gate

Gate	Trace	Function	Format
DMDM	RM 0101 RM 0102	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

You can find descriptions of these functions and their input and output parameters, in the section. dealing with the corresponding generic format, in Chapter 78, “Domain manager domain (DM),” on page 663.

In Initialization processing, the Recovery Manager
- Obtains initialization parameters from Parameter Manager,
- Determines the type of start to be performed,
- Processes it's data from the Global Catalog,
- Processes recovery information from the System Log.

In Quiesce processing, the Recovery Manager takes the warm keypoint.

Recovery Manager domain’s call back formats

Table 87 describes the call back format owned by the Recovery Manager domain and shows the function performed on the calls.

Table 87. Call back format owned by the Recovery Manager domain

Format	Calling module	Function
RMRO	DFHRMUO DFHRMUP DFHRMUQ DFHRMUW DFHRMUO DFHRMRO2 DFHRMRO3 DFHRMRO4 DFHRMROS DFHRMROU	PERFORM_COMMIT PERFORM_PREPARE START_BACKOUT DELIVER_BACKOUT_DATA END_BACKOUT PERFORM_SHUNT PERFORM_UNSHUNT
RMDE	DFHRMR1S DFHRMR1D DFHRMR1E DFHRMR1D	START_DELIVERY DELIVER_RECOVERY END_DELIVERY DELIVER_FORGET
RMKP	DFHRMR1K	TAKE_KEYPOINT
RMLK	DFHRMLSP DFHRMLSP DFHRMLSD DFHRMLSD DFHRMLSO DFHRMLSS DFHRMLSU	PERFORM_PRELOGGING PERFORM_PREPARE REPLY_DO_COMMIT SEND_DO_COMMIT PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT

In the descriptions of the formats that follow, the “input” parameters are input not to Recovery Manager domain, but to the domain being called by the Recovery Manager. Similarly, the “output” parameters are output by the domain that was called by Recovery Manager domain, in response to the call.

RMRO gate, PERFORM_COMMIT function

This function requires the Recovery Manager client to perform phase 2 of syncpoint processing.

Input parameters

WORK_TOKEN
The Recovery Manager client’s work token for the syncpointing unit of work.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES|NO

UOW_STATUS
The status of the current unit of work. It can have any one of these values:

FORWARD|BACKWARD

RESTART An optional parameter specifying whether a backing out transaction will be restarted. It can have any one of these values:

YES|NO

Output parameters

FORGET_RECORD
A value specifying whether all obligations to this Recovery Manager client have been discharged. It can have any one of these values:

YES|NO

RESPONSE is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMRO gate, PERFORM_PREPARE function

This function requires the Recovery Manager client to perform phase 1 of syncpoint processing.

Input parameters

WORK_TOKEN The Recovery Manager client's work token for the syncpointing unit of work.

CONTINUE A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES|NO

Output parameters

VOTE A value specifying the Recovery Manager client's vote on the outcome of the syncpointing unit of work. It can have any one of these values:

YES|NO|NO_CONTINUE|READ_ONLY

RESPONSE is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMRO gate, START_BACKOUT function

This function notifies the Recovery Manager client that backout processing is about to be performed for the unit of work.

Input parameters

WORK_TOKEN The Recovery Manager client's work token for the syncpointing unit of work.

CONTINUE A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES|NO

REMOVE A parameter specifying whether or not the backout is due to an invocation of the REMOVE function of the RMRE gate. It can have any one of these values:

YES|NO

Output parameters

RESPONSE is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMRO gate, DELIVER_BACKOUT_DATA function

This function requires the Recovery Manager client process backout data from the system log for the unit of work.

Input parameters

WORK_TOKEN The Recovery Manager client's work token for the syncpointing unit of work.

DATA
A buffer containing the data previously logged with BACKWARD_DATA(YES) via the APPEND function of the RMRE gate.

RESOURCE_ID
An optional parameter specifying the name of the resource with which the logged data is associated.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES|NO

FORWARD_DATA
A parameter specifying whether or not the data was originally logged as FORWARD_DATA. It can have any one of these values:

YES|NO

REMOVE
A parameter specifying whether or not the backout is due to an invocation of the REMOVE function of the RMRE gate. It can have any one of these values:

YES|NO

CLUSTER_ID
A buffer to receive a symbolic name identifying the resource.

LOCAL_ACCESS_ID
A buffer to receive the specific name of the resource

Output parameters

KEEP
A value specifying whether the backout action failed, implying the record should be kept and not forgotten. It can have any one of these values:

YES|NO

RESPONSE
is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMRO gate, END_BACKOUT function

This function notifies the Recovery Manager client that backout processing has completed for the unit of work.

Input parameters

WORK_TOKEN
The Recovery Manager client's work token for the syncpointing unit of work.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES|NO

REMOVE
A parameter specifying whether or not the backout is due to an invocation of the REMOVE function of the RMRE gate. It can have any one of these values:

YES|NO

Output parameters

RESPONSE
is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMRO gate, PERFORM_SHUNT function

This function notifies the Recovery Manager client that the unit of work is about to shunt.

Input parameters

WORK_TOKEN
The Recovery Manager client's work token for the syncpointing unit of work.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES|NO

Output parameters

NEXT_WORK_TOKEN
A value for the Recovery Manager client's work token in the following unit of work.

RESPONSE is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMRO gate, PERFORM_UNSHUNT function

This function notifies the Recovery Manager client that the unit of work is unshunting.

Input parameters

WORK_TOKEN The Recovery Manager client's work token for the syncpointing unit of work.

Output parameters

RESPONSE is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMDE gate, START_DELIVERY function

This function notifies the Recovery Manager client that system recovery processing is about to be performed.

Input parameters

None

Output parameters

RESPONSE is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMDE gate, DELIVER_RECOVERY function

This function requires the Recovery Manager client to process recovery data from the system log.

Input parameters

RESOURCE_ID An optional parameter specifying the name of the resource with which the logged data is associated.

DATA A buffer containing the data previously logged with BACKWARD_DATA(YES) via the APPEND function of the RMRE gate.

FORWARD_DATA A parameter specifying whether or not the data was originally logged as FORWARD_DATA. It can have any one of these values:

YES|NO

BACKWARD_DATA

A parameter specifying whether or not the data was originally logged as BACKWARD_DATA. It can have any one of these values:

YES|NO

KEYPOINT A parameter specifying whether or not the data was logged as part of a keypoint. It can have any one of these values:

YES|NO

BACKED_OUT A parameter specifying whether or not the update the data is associated with backed out. It can have any one of these values:

YES|NO

UOW A parameter specifying whether the data is related to a particular unit of work. It can have any one of these values:

YES|NO

UOW_STATUS An optional parameter specifying the status of unit of work the data belongs to (if any). It can have any one of these values:

FORWARD|BACKWARD|IN_DOUBT|IN_FLIGHT

LOCAL_UOW_ID An optional parameter specifying the local UOWID of the unit of work the data belongs to (if any).

Output parameters

RESPONSE is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMDE gate, END_DELIVERY function

This function notifies the Recovery Manager client that all recovery information from the system log has been processed.

Input parameters

None

Output parameters

RESPONSE is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMDE gate, DELIVER_FORGET function

This function notifies the Recovery Manager client that FORGET processing is required for some resource in a unit of work.

Input parameters

LOCAL_ACCESS_ID
A parameter specifying the name of the resource associated with the forget processing.

UOW It can only have the value YES.

UOW_STATUS The status of the unit of work. It can have any one of these values:

FORWARD|BACKWARD|IN_DOUBT|IN_FLIGHT

LOCAL_UOW_ID The local UOWID of the unit of work.

Output parameters

RESPONSE is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMKP gate, TAKE_KEYPOINT function

This function requires the Recovery Manager client to perform keypoint processing.

Input parameters

SHUTDOWN A parameter specifying whether the keypoint is the warm keypoint taken during shutdown or an activity keypoint. It can have any one of these values:

YES|NO

Output parameters

RESPONSE is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMLK gate, PERFORM_PRELOGGING function

This function notifies the Recovery Manager client that phase 1 of syncpoint processing is about to occur.

Input parameters

RMC_TOKEN The Recovery Manager client's token associated with the Recovery Manager Link object.

INITIATOR
A parameter specifying whether the remote system is the initiator of the syncpoint. It can have any one of these values:

YES|NO

COORDINATOR(YES]NO)
A parameter specifying whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:

YES|NO

Output parameters

RESPONSE
is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMLK gate, PERFORM_PREPARE function

This function requires the Recovery Manager client perform phase 1 of syncpoint processing.

Input parameters

RMC_TOKEN
The Recovery Manager client's token associated with the Recovery Manager Link object.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES|NO

SYSTEM
A parameter specifying whether PERFORM_PREPARE call is part of a syncpoint or the result of EXEC CICS ISSUE PREPARE. It can have any one of these values:

YES|NO

RECOVERY_STATUS
A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system. It can have any one of these values:

NECESSARY|UNNECESSARY|SYNC_LEVEL_1

Output parameters

VOTE
A value specifying the Recovery Manager client's vote on the outcome of the syncpointing unit of work. It can have any one of these values:

YES|NO|NO_CONTINUE|READ_ONLY|HEURISTIC_MIXED

RESPONSE
is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMLK gate, REPLY_DO_COMMIT function

This function requires the Recovery Manager client communicate the result of this systems phase 1 syncpoint processing to the coordinating system, and obtain the outcome of the distributed unit of work.

Input parameters

RMC_TOKEN
The Recovery Manager client's token associated with the Recovery Manager Link object.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES|NO

SINGLE_UPDATER
A parameter specifying whether the single updater optimization is being performed. It can have any one of these values:

YES|NO

Output parameters

ACCESSIBLE
A value specifying whether communication with the remote system failed. It can have any one of these values:

YES|NO|SHUNTED

VOTE
A value specifying the outcome of the syncpointing unit of work. It can have any one of these values:

YES|NO|NO_CONTINUE|READ_ONLY|HEURISTIC_MIXED

RESPONSE
is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMLK gate, SEND_DO_COMMIT function

This function requires the Recovery Manager client communicate the result of this systems phase 1 syncpoint processing to the last agent system, and obtain the outcome of the distributed unit of work.

Input parameters

RMC_TOKEN
The Recovery Manager client's token associated with the Recovery Manager Link object.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES|NO

SINGLE_UPDATER
A parameter specifying whether the single updater optimization is being performed. It can have any one of these values:

YES|NO

Output parameters

ACCESSIBLE
A value specifying whether communication with the remote system failed. It can have any one of these values:

YES|NO|SHUNTED

VOTE
A value specifying the outcome of the syncpointing unit of work. It can have any one of these values:

YES|NO|NO_CONTINUE|READ_ONLY|HEURISTIC_MIXED

RESPONSE
is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMLK gate, PERFORM_COMMIT function

This function requires the Recovery Manager client perform phase 2 of syncpoint processing.

Input parameters

RMC_TOKEN
The Recovery Manager client's token associated with the Recovery Manager Link object.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES|NO

SINGLE_UPDATER
A parameter specifying whether the single updater optimization is being performed. It can have any one of these values:

YES|NO

UOW_STATUS
The status of the syncpointing unit of work. It can have any one of these values:

FORWARD|BACKWARD

RESTART
An optional parameter specifying whether a backing out transaction will be restarted. It can have any one of these values:

YES|NO

COORDINATOR
A parameter specifying whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:

YES|NO

INITIATOR
A parameter specifying whether the remote system is the initiator of the syncpoint. It can have any one of these values:

YES|NO

PRESUMPTION
A parameter specifying whether the remote system assumes the presume abort or presume nothing protocols. It can have any one of these values:

ABORT|NOTHING

RECOVERY_STATUS
A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system. It can have any one of these values:

NECESSARY|UNNECESSARY|SYNC_LEVEL_1

Output parameters

ACCESSIBLE
A parameter specifying that the communications link to the remote system has failed. It can have any one of these values:

YES|NO|SHUNTED

FORGET
A parameter specifying whether all obligations to the remote system with respect to recovery have been discharged. It can have any one of these values:

YES|NO

PASS
A parameter specifying whether an equivalent Recovery Manager Link object should be created in the following unit of work. It can have any one of these values:

YES|NO

ABEND
A parameter specifying whether an abend occurred during the PERFORM_COMMIT call-back. It can have any one of these values:

YES|NO

NEXT_RECOVERY_STATUS
A parameter specifying the initial RECOVERY_STATUS of the Recovery Manager Link object created in the following unit of work as a result of PASS(YES). It can have any one of these values:

NECESSARY|UNNECESSARY|SYNC_LEVEL_1|DEFAULT

RESPONSE
is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMLK gate, PERFORM_SHUNT function

This function notifies the Recovery Manager client that the unit of work is shunting.

Input parameters

RMC_TOKEN
The Recovery Manager client's token associated with the Recovery Manager Link object.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES|NO

RECOVERY_STATUS
A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system. It can have any one of these values:

NECESSARY|UNNECESSARY|SYNC_LEVEL_1

Output parameters

FORGET
A parameter specifying whether all obligations to the remote system with respect to recovery have been discharged. It can have any one of these values:

YES|NO

RESPONSE
is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMLK gate, PERFORM_UNSHUNT function

This function notifies the Recovery Manager client that the unit of work is unshunting.

Input parameters

LINK_TOKEN
A token identifying the Recovery Manager Link object to be unshunted.

LOGNAME_BUFFER
A parameter specifying a buffer containing the logname of the remote system.

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of the remote system, or the name of the External Resource Manager.

LINK_ID_BUFFER
A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier.

LINK_ID_SOURCE
An optional parameter specifying whether the local or remote system allocated the session. It can have any one of these values:

LOCAL|REMOTE

Output parameters

RESPONSE
is the Recovery Manager client's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

Modules

Module	Function
DFHRMCD	Handles the functions of the RMCD gate.
DFHRMCD1	Initialises the Client Directory Class.
DFHRMCD2	Quiesces the Client Directory Class.
DFHRMCI2	Sets the callback gate of a Recovery Manager client.
DFHRMCI3	Waits for a registered Recovery Manager client to set its callback gate.
DFHRMCI4	Waits for a registered Recovery Manager client to set its callback gate and calls it with a given parameter list.
DFHRMDM	Recovery Manager domain initialization and termination. Handles the DMDM and RMDM gate functions.
DFHRMUTL	Recovery Manager batch utility.
DFHRMDU0	Formats the Recovery Manager control blocks.
DFHRMDU2	Starts a browse of all Recovery Manager client work tokens during dump formatting.
DFHRMDU3	Gets the next Recovery Manager client work token during dump formatting.
DFHRMDU4	Ends a browse of all Recovery Manager client work tokens during dump formatting.
DFHRMLK1	Initialises the Recovery Manager Link Class.
DFHRMLK2	Handles the INITIATE_RECOVERY function of the RMLN gate.
DFHRMLK3	Inquires whether a Logname is in-use by any Recovery Manager Link.
DFHRMLK4	Handles the CLEAR_PENDING function for a particular Recovery Manager Link.
DFHRMLK5	Collects statistics from the Recovery Manager Link Class.
DFHRMLKQ	Quiesces the Recovery Manager Link Class.
DFHRMLN	Handles the functions of the RMLN gate.
DFHRMLSD	Asks the coordinator Recovery Manager Link to decide the outcome of the unit of work.

Module	Function
DFHRMLSF	Determines the reason for a unit of work being in doubt.
DFHRMLSO	Commits the Recovery Manager Links for a unit of work.
DFHRMLSP	Prepares the Recovery Manager Links for a unit of work.
DFHRMLSS	Shunts the Recovery Manager Links for a unit of work.
DFHRMLSU	Unshunts the Recovery Manager Links for a unit of work.
DFHRML1D	Reconstructs Recovery Manager Links from log records.
DFHRMNM	Handles the functions of the RMNM gate.
DFHRMNM1	Initialises the Recovery Manager Lognames Class.
DFHRMNS1	Initialises the Recovery Manager Logname Set Class.
DFHRMNS2	Quiesces the Recovery Manager Logname Set Class.
DFHRMOFI	Initialises a Recovery Manager Object Factory.
DFHRMRO	Handles the functions of the RMRO gate.
DFHRMROO	Handles FORGET processing for Recovery Manager Resource Owners.
DFHRMROS	Shunts a Recovery Manager Resource Owner.
DFHRMROU	Unshunts a Recovery Manager Resource Owner.
DFHRMROV	Handles AVAIL processing for Recovery Manager Resource Owners.
DFHRMRO1	Initialises the Recovery Manager Resource Owner Class.
DFHRMRO2	Signals start_backout to a Recovery Manager Resource Owner.
DFHRMRO3	Delivers backout data to a Recovery Manager Resource Owner.
DFHRMRO4	Signals end_backout to a Recovery Manager Resource Owner.
DFHRMR1D	Delivers recovery data to a Recovery Manager Resource Owner.
DFHRMR1E	Signals end of recovery to a Recovery Manager Resource Owner.
DFHRMR1K	Signals a keypoint to a Recovery Manager Resource Owner.
DFHRMR1S	Signals start of recovery to a Recovery Manager Resource Owner.
DFHRMSL	Handles the functions of the RMSL gate.
DFHRMSLF	Forces the System Log.
DFHRMSLJ	Checks for Chain independence during recovery.
DFHRMSLL	Closes a Chain on the System Log.
DFHRMSLO	Opens a Chain on the System Log.
DFHRMSLV	Moves a Chain on the System Log.
DFHRMSLW	Writes a record to a Chain on the System Log.
DFHRMSL1	Initialises the Recovery Manager System Log Class.
DFHRMSL2	Starts a browse of a Chain on the System Log.
DFHRMSL3	Reads a Record from a Chain on the System Log.
DFHRMSL4	Ends a browse of a Chain on the System Log.
DFHRMSL5	Performs restart processing for Recovery Manager System Log Class.
DFHRMSL6	Schedules keypoint activity.
DFHRMSL7	Performs keypoint processing.
DFHRMST	Handles STST functions for Recovery Manager.
DFHRMST1	Initializes the Recovery Manager Statistics Class.
DFHRMTRI	Formats Recovery Manager trace entries.

Recovery Manager Domain (RM)

Module	Function
DFHRMUC	Creates a RMUW (unit of work) object.
DFHRMUO	Commits a unit of work.
DFHRMUW	Handles the functions of the RMUW gate.
DFHRMUWB	Handles data during backout of a unit of work.
DFHRMUWE	Handles activities when a unit of work is unshunted.
DFHRMUWF	Forces log records for a unit of work.
DFHRMUWH	Holds an RMUW object.
DFHRMUWJ	Forces a unit of work to take a unilateral decision.
DFHRMUWL	Handles notification that all remote remotes have finished processing.
DFHRMUWN	Schedules a unit of work to be unshunted.
DFHRMUWP	Handles notification that a local resource has become available.
DFHRMUWQ	Handles commit or backout of an unshunted, in doubt unit of work.
DFHRMUWS	Records the outcome of a unit of work during resynchronization.
DFHRMUWU	Records the local LU name.
DFHRMUWV	Handles notification that a local resource has become available.
DFHRMUWW	Writes a record belonging to a unit of work to the System Log.
DFHRMUW0	Releases an RMUW object.
DFHRMUW1	Initializes the Recovery Manager Unit of Work Class.
DFHRMUW2	Collects the Recovery Manager Unit of Work Class Statistics.
DFHRMUW3	Handles the INQUIRE_UOW_TOKEN function.
DFHRMU1C	Sets the Chain token for a unit of work.
DFHRMU1D	Handles log records of units of work during recovery.
DFHRMU1E	Signals that all records have been recovered from the System Log during recovery.
DFHRMU1F	Handles an in doubt wait timeout.
DFHRMU1J	Inquires whether all unit of work chains are disjoint.
DFHRMU1K	Keypoints a unit of work.
DFHRMU1L	Handle XMPP_FORCE_PURGE_INHIBIT_QUERY.
DFHRMU1N	Handle XMPP_FORCE_PURGE_INHIBIT_QUERY.
DFHRMU1Q	Handle the NOTIFY function of the TISR gate.
DFHRMU1R	Performs restart processing for Recovery Manager Unit of Work Class.
DFHRMU1S	Signals that recovery of log records is about to be performed.
DFHRMU1U	Process a unit of work after recovery.
DFHRMU1V	Requests time out interval notification for a unit of work.
DFHRMU1W	Cancels wait time out notification for a unit of work.
DFHRMVP1	Initializes the Recovery Manager Variable Length Subpool Class.
DFHRMXNE	Reattaches a transaction to process an unshunted unit of work.
DFHRMXN2	Schedules a keypoint.
DFHRMXN3	The keypoint program.
DFHRMXN4	Restarts the Recovery Manager Transaction Class.
DFHRMXN5	Increments Recovery Manager statistics for a Transaction.

Exits

None

Trace

The point IDs for the Recovery Manager domain are of the form RM xxxx the corresponding trace levels are RM 1, RM 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 100. RRMS domain (RX)

The RRMS domain is responsible for managing interaction with OS/390 Recoverable Resource Management Services (RRMS) and in particular, Resource Recovery Services (RRS) which is a component of RRMS.

RRMS domain's specific gates

Table 88 summarizes the RX domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 88. RX domain's specific gate

Gate	Trace	Function	XPI
RXDM	RX 0101 RX 0102	INQUIRE_RRS SET_PARAMETERS	NO NO
RXUW	RX 0401 RX 0402	PUT_CLIENT_REQUEST GET_CLIENT_REQUEST INQUIRE	NO NO NO

RXDM gate, INQUIRE_RRS function

The INQUIRE_RRS function of the RXDM gate is used to determine the status of CICS's interface with OS/390 Recoverable Resource Management Services (RRMS).

Output Parameters

OPEN
Returns YES or NO to indicate if the interface with RRMS is open.

[RESTART_STATE]
Returns a value to indicate the state of restart processing with Resource Recovery Services (RRS). One of these values is returned:

NOT_STARTED
Restart processing has not started

STARTING
Restart is in progress

COLD
Restart processing is complete, and RRS was cold started.

WARM
Restart processing is complete, and RRS was warm started.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RXDM gate, SET_PARAMETERS function

The SET_PARAMETERS function of the RXDM gate is used to pass the values of relevant System Initialization parameters to the domain.

Input Parameters

RRMS
Specifies the value of the RRMS System Initialization Parameter. It can have one of these values:

YES|NO

Output Parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

RXUW gate, PUT_CLIENT_REQUEST function

The PUT_CLIENT_REQUEST function of the RXDM gate is used to associate a request from a client with an RRS Unit of Recovery (UR).

Input parameters

TRANSACTION_ID
The transaction id associated with the request. This parmeter is used to correlate succesive requests for the same transaction instance.

USERID
The userid associated with the request. This parmeter is used to correlate succesive requests for the same transaction instance.

CONNECTION
The connection on which the client request was received. This parameter is used to identify the source of the request in any messages that are issued.

CONTEXT_TOKEN
The token representing the RRMS context for which the request is issued.

URID
The identifier of the RRS Unit of Recovery associated with the context.

PASS_TOKEN
A token used to protect against unauthorised use of the context token and URID.

CLIENT_TOKEN
A token representing the client of the UR.

CLIENT_TYPE
Indicates the type of client of the transaction. The only permissible value is TERMINAL

Output parameters

NEW_UR
Indicates whether a new UR has been created for this request. It can have one of these values:

YES
indicates that a new UR has been created

NO_AND_READY
indicates that the request was associated with an existing UR and that task is ready to receive the request.

NO_AND_NOT_READY
indicates that the request was associated with an existing UR but that task is not ready to receive the request. This usually occurs when the original request has timed out and another transactional EXCI request in the same RRS UR has been sent by the EXCI job.

NO_AND_NOTASK
indicates that the request was associated with an existing UR but that task has not yet expressed an interest in the UR. This can occur when the original request has been held by MAXTASK or TRANCLASS (TCLASS) limits and has timed out, and another Transactional EXCI request in the same UR has been sent by the EXCI job.

UR_TOKEN
is the token by which the UR associated with the request is known by the RX domain.

TRANSACTION_NUMBER
The transaction number of the transaction associated with the request.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

RXUW gate, GET_CLIENT_REQUEST function

The GET_CLIENT_REQUEST function of the RXDM gate is used to suspend a transaction until the PUT_CLIENT_REQUEST is issued for the same Unit of Recovery.

Input parameters

UR_TOKEN
is the token by which the UR associated with the request is known by the RX domain.

[TIMEOUT]
The time (in seconds) for which the transaction should be suspended. If this paramter is omitted, the transaction will be suspended indefinitely.

Output paramters

CLIENT_TOKEN
A token representing the client of the UR.

CLIENT_TYPE
Indicates the type of client of the transaction. The only possible value is TERMINAL

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are

SYNCPOINT
RRS has requested a syncpoint

BACKOUT
RRS has requested rollback

RACE
RRS has requested syncpoint or rollback and a client request has been received at the same time

[REASON] is also returned when RESPONSE is PURGED. Possible values are

TASK_CANCELLED
The task has been purged

TIMED_OUT
The request has timed out

RXUW gate, INQUIRE function

The INQUIRE function requests attributes of a Unit of Recovery

Input parameters

UR_TOKEN is the token which identifies the Unit of Recovery

Output parameters

[URID] The identifier of the Unit of Recovery used by RRMS.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

Modules

Module	Function
DFHRXDM	RX domain management and global functions.
DFHRXUW	RX domain unit-of-work related functions.
DFHRXSVC	RX domain SVC code for RRMS authorized interface.
DFHRXXRG	RX domain Registration Services exits.
DFHRXXRM	RX domain Resource Manager exits.
DFHRXDUF	RX domain dump formatting.
DFHRXTRI	RX domain trace interpretation.

Exits

None

Trace

The point IDs for the RRMS domain are of the form RX xxxx the corresponding trace levels are RX 1, RX 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 101. RequestStream domain (RZ)

The RequestStream domain provides connectivity between elements of the Corbaserver and EJB components in a sysplex to allow transfer of GIOP requests from a requester to a request processor, and to permit appropriate workload balancing of the deployment of those requests.

It provides basic management functions for RequestStreams:
- Create RequestStream
- Destroy RequestStream
- Publicise RequestStream (see PublicId)
- Join requeststream
- Leave RequestStream
- Send/receive requests on RequestStreams

The RequestStream domain also manages *Transports*, which are single communication mechanisms that transfer bytes from one task to another in the sysplex. They occur in different types, depending upon which technology is appropriate for the route that needs to be taken. For example, there are “InStore” transports, “MRO” transports and “Socket” transports.

RequestStream domain’s specific gates

Table 89 summarizes the RequestStream domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, and the functions provided by the gates. None of the functions are available through the exit programming interface (XPI).

Table 89. RequestStream domain specific gates

Gate	Trace	Function	XPI
RZSO	RZ 0110 RZ 0111	CREATE SEND_REQUEST RECEIVE_REPLY LEAVE JOIN IS_ID_LOCAL	NO NO NO NO NO NO
RZTA	RZ 0120 RZ 0121	RECEIVE_REQUEST SEND_REPLY GET_PUBLIC_ID GET_SERVER_DATA GET_CURRENT GET_JOIN_DATA TERMINATE	NO NO NO NO NO NO NO
RZRT	RZ 0170 RZ 0171	SET_EXIT_PROGRAM	NO
RZRJ	RZ 0180 RZ 0181	PERFORM_JOIN	NO

RZSO gate, CREATE function

Create a RequestStream and return a (local region) source RequestStream token for it.

The target process(or) is identified *either* by **USERID** and **TRANID** *or* by **HOST_IP_ADDRESS** and **PORT_NUMBER**. Precisely one of these groups must be provided.

(The **HOST_IP_ADDRESS** is a character string as expected by the internal sockets domain interfaces.)

The **SERVER_DATA** may be retrieved at the target (**RZTA**) interface and is copied (and fixed) on this call.

The response is **(exception, service_not_available)** if it is not possible to resolve the target, or to set up a connection to the target. (Success does not guarantee that this exception will not occur on the **SEND** function.)

The response is **(exception, target_unknown)** if the **HOST_IP_ADDRESS** character string is malformed (as detected by the sockets domain interfaces).

The response is **invalid** when the parameters are badly formed, in particular if there is not the right combination of target identification parameters.

Input parameters

[USERID]
Userid under which the requests are to be processed.

[TRANID]
TranId of the transaction which runs the target processor.

[HOST_IP_ADDRESS]
Identification of the target which is to process the requests.

[PORT_NUMBER]
Further identification of the target.

[SERVER_BLOCK]
Data associated with the RequestStream available at the target end by the server using the **RZTA** interface.

[SSL_REQUIRED]
Values: **YES, NO**

Whether to use SSL on a socket transport. Otherwise ignored.

[CERTIFICATE_NAME]
Further information for an SSL socket transport. Otherwise ignored.

Output parameters

RS_TOKEN
Token by which RequestStream is identified on all subsequent requests from this task on this region.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVICE_NOT_AVAILABLE TARGET_UNKNOWN

RZSO gate, SEND_REQUEST function

The source RequestStream token and the request (coded as a **RUEI** or as a contiguous data block) is passed as input. Either a ruei or a block must be used, not both. If this is not so then an **invalid** response is returned.

The request is deemed to be entire and may be presented to the target. Data may be transported across the transport mechanism during this call. The request may be of zero length, this does *not* imply that nothing is transported.

If the source RequestStream token does not exist (in the local region) the response **(exception, rs_token_unknown)** is returned.

If a transport mechanism fails to respond, or is not functional, then the response **(exception, service_not_available)** is returned. If it fails during transmission then **(exception, transport_failure)** is returned. The distinction is that in the former case there is no transport mechanism and in the latter there is still one (albeit inoperational).

Input parameters

RS_TOKEN
Token returned on CREATE by which RequestStream is identified.

[REQUEST_RUEI]
Reusable-extended-Iliffe Vector which describes contiguous bytes to send as a request, supplied in possibly discontiguous blocks. Exclusive with REQUST_BLOCK.

[REQUEST_BLOCK]
Request data to send described as a single block. Exclusive with REQUEST_RUEI.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	RS_TOKEN_UNKNOWN TRANSPORT_FAILURE SERVICE_NOT_AVAILABLE

RZSO gate, RECEIVE_REPLY function

A reply is returned (blocks until one is available).

The "reply_buffer" is set with as much data as will fit. The total number of bytes possible to be transferred is returned in the buffer as well as as many bytes of the reply that will fit. The "reply_buffer" size +is not set+ although the number of bytes to be set in the buffer is set as usual. The call can be re-issued before any other calls to the RequestStreams source interface (using this token) to receive the remaining reply bytes. Reply data is not redelivered.

The final call to "receive_reply" indicates that all of the data to be transferred fit within the buffer passed. The final call to "receive_reply" will allow the transport to change direction, which means that further calls to "receive_reply" will be an error and will give a response "(exception, transport_failure)". If a "notify" callback has been called before this function is issued then "receive_reply" will be satisfied. The status of the "notify" callback is not affected by this call ("notify" is disabled automatically when it is issued) except that if there is data to deliver, and a "notify" callback is enabled with "listen", and has not been called, then the "notify" callback will be called as part of the processing of this function. This can be avoided, if necessary, by issuing a "cancel" beforehand.

If the RequestStream token is not known then the response is "(exception, rs_token_unknown)". If the transport service is not open then the response "(exception, service_not_available)" is returned. If the transport service is still open but gives some sort of error (for example, is not in the correct state to receive a reply) then the response is "(exception, transport_failure)". If the request processor cannot, for some reason, process the request, and returned an exception, or failed during execution, then the response is "(exception, request_processor_failure)". Processor failure, when it can be ascertained, takes precedence over transport failure.

Input parameters

RS_TOKEN
Token returned on CREATE by which RequestStream is identified.

[REPLY_BUFFER]
Buffer in which reply bytes are assembled.

[MINIMUM_DATA_LENGTH]
Minimum amount of data to accept (multiple transfers may occur until this amount is received).

Output parameters

[REPLY_DATA_LENGTH]
Total length of reply (even if not all received in one call).

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	RS_TOKEN_UNKNOWN SERVICE_NOT_AVAILABLE TRANSPORT_FAILURE REQUEST_PROCESSOR_FAILURE

RZSO gate, LEAVE function

Remove this source from its RequestStream. The RequestStream is modified so that the "rs_token" (which must denote a source end of the RequestStream) is no longer valid. (A token value may or may not be reissued by "RZ" on another "create" or "join" request - however the caller must not rely on its value after "leave".)

If the RequestStream "rs_token" is valid but does not denote a source end of a RequestStream known in this region the response "(exception, rs_token_not_source)" is returned.

If the transport mechanism fails then the response "(exception, transport_failure)" is returned, +however the RequestStream source token is still invalidated+.

If a "notify" is enabled for this RequestStream token (with a "listen" call) then the "notify" callback is called with the "CLOSE" parameter prior to removing the RequestStream source.

Input parameters

RS_TOKEN The token returned on CREATE identifying the RequestStream for this region.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	RS_TOKEN_UNKNOWN TRANSPORT_FAILURE RS_TOKEN_NOT_SOURCE

RZSO gate, JOIN function

"Join" the RequestStream identified by the "public_id".

If the required transport mechanism is not available, or fails in use, the appropriate exception is returned as for "create".

If the RequestStream, identified by the "public_id", does not exist (because the target end does not exist) then this call does not detect this. Instead a new request processor will be created implicitly just as for "create".

The "userid" (if supplied) must match that used on the "create", otherwise an error may occur later in (Request Processor) processing. This is not detected at this call. The "tranid" and the "server_data" is supplied in case the RequestStream is recreated on this call, otherwise they are ignored. They may be omitted as in *create*.

If the "public_id" is not valid, or cannot be interpreted then the response "(exception, public_id_invalid)" will be returned.

The "rs_token" for the local source RequestStream is returned as result.

Input parameters

PUBLIC_ID
Public RequestStream Identifier -- valid for all participating regions in the logical server -- of the target RequestStream, which may be in a separate region.

[USERID]
The security userid under which the target RequestStream should be executing.

[TRANID]
The transaction identifier for the target RequestStream task.

[SERVER_BLOCK]
The server data that was specified. This parameter and Tranid are used only if a new target RequestStream needs to be created owing to the omission of the target RequestStream processor.

Output parameters

RS_TOKEN
The local region token by which the source RequestStream (that connects to the target) is known on all subsequent calls.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVICE_NOT_AVAILABLE TRANSPORT_FAILURE PUBLIC_ID_INVALID

RZSO gate, IS_ID_LOCAL function

Return "yes" if the "public_id" refers to the local region as the target region for the RequestStream. Otherwise "no".

This is a non-blocking call that gives the response "(exception, public_id_invalid)" if the "public_id" is detectably invalid (e.g. has the wrong format) and otherwise has no exceptions.

Input parameters

PUBLIC_ID
Public RequestStream Identifier -- valid for all participating regions in the logical server -- of a target RequestStream, which may be in a separate region.

Output parameters

LOCAL
Values:

YES|NO indicating whether the identified PUBLIC_ID is in the local region or not. There is no guarantee that it exists in either case.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	PUBLIC_ID_INVALID

RZTA gate, RECEIVE_REQUEST function

Get the next request. This call blocks if there is no request ready, and returns when a request becomes available or if the RequestStream is destroyed while waiting ("terminate"d). This call will be satisfied without undue waiting if a "notify" callback has been invoked.

A request is delivered by means of a buffer. The total length of the buffer data available is placed in the buffer content value, and as many bytes as will fit are copied to the buffer data area. If there are more bytes than will fit in the buffer area then another receive request may be issued to receive more, and more will be copied until all are delivered.

Should a request not be fully delivered on this interface then the underlying transport will not be reversed (as in half-duplex communications) until another receive is issued. Only when all of the request is delivered will the request be deemed to be delivered and the underlying transport be able to flip.

Should other interface calls be issued on this RequestStream (target) then data bytes of this request may be lost. The request data are bound to the task which is issuing the receive request commands, and the task cannot change in the middle of receiving a request.

The "correlation_id" (optionally received and identical on all the calls for one request) identifies the source from which the request comes and is required when sending a reply on this RequestStream (with "send_reply"). It is guaranteed distinct for each distinct source of this RequestStream. The reply to this request (if there is one) should be accompanied by the same "correlation_id" (on "send_reply") otherwise no guarantee can be made that the reply will return to the correct source.

Input parameters

REQUEST_BUFFER
: Buffer into which the request is received.

[MINIMUM_DATA_LENGTH]
: The minimum data length that should secure a response if not all the request is received. Multiple transfers may occur if not enough data is available when the request is issued.

Output parameters

[CORRELATION_ID]
: The identifier of the requester using this RequestStream. It is used when replying to this request (using SEND_REPLY on this RequestStream) so as to identify the source from which the request was issued. It is valid only while this RequestStream is available to this transaction.

RESPONSE
: is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
: Returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	REQUESTSTREAM_NOT_CURRENT SERVICE_NOT_AVAILABLE TRANSPORT_FAILURE
INVALID	INVALID_BUFFER

RZTA gate, SEND_REPLY function

Send a reply to a source identified by "correlation_id".

The "correlation_id" must be one returned by the "receive_request" function for the current RequestStream, or else the exception "correlation_id_unknown" may be returned.

A reply may consist of the empty sequence of bytes in which case an empty reply is sent.

The usual exceptions are returned for transportation failures.

Input parameters

REPLY_BLOCK
: A block containing the complete contiguous reply.

CORRELATION_ID
The correlation id received on RECEIVE_REQUEST for the request to which this is the reply.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	REQUESTSTREAM_NOT_CURRENT CORRELATION_ID_UNKNOWN SERVICE_NOT_AVAILABLE TRANSPORT_FAILURE

RZTA gate, GET_SERVER_DATA function

Return the server data for the current RequestStream.

If there is a current RequestStream then server data are available. If there is no current RequestStream the response "(exception, requeststream_not_current)" is returned. If the server data are known but empty (zero-length sequence of bytes), then the outputs are set to indicate zero-length content.

The "server_block" is input so that it may be filled on return. If the data area is too small then the response "(exception, server_block_too_small)" is returned +and no data is transferred+. In this case the output "server_data_length" is set to indicate the total number of bytes in the server data.

Server data do not change while the RequestStream remains current. (A RequestStream may be recreated as a result of a "join" if it has previously been destroyed or terminated. At this point the server data may be different from the original values.)

Input parameters

SERVER_BLOCK Block in which the server data is placed on output.

Output parameters

SERVER_DATA_LENGTH
The number of bytes of the server data, even if not all were returned.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	REQUESTSTREAM_NOT_CURRENT SERVER_BLOCK_TOO_SMALL

RZTA gate, GET_PUBLIC_ID function

The public identifier of the RequestStream for the current transaction is returned. (If the target of the RequestStream is not internal to the plex there may not be a public identifier, for example in the case of outbound RequestStreams. In this case the response is "(exception, public_id_unknown)". However, this should never happen on this interface, since such a RequestStream will never be set in the "RZ" transaction manager token for a transaction instance.)

The response "(exception, RequestStream_not_current)" is returned if the XM token is not valid in the local region, or if no XM token for the RequestStream is set.

Output parameters

PUBLIC_ID
Public RequestStream Identifier -- valid for all participating regions in the logical server -- of the current target RequestStream which must be attached to this task/transaction.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	REQUESTSTREAM_NOT_CURRENT PUBLIC_ID_UNKNOWN

RZTA gate, GET_CURRENT function

The token for the RequestStream for the current transaction is returned. If the "XM" token is not set, or is set to an invalid value, then the response "(exception, RequestStream_not_current)" is returned.

Output parameters

RS_TOKEN
Token for current target RequestStream which must be attached to this task.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	REQUESTSTREAM_NOT_CURRENT

RZTA gate, GET_JOIN_DATA function

This is a utility function used by the join task which can thereby reduce the number of domain calls to RZ when acting as intermediary to another task on remote join.

If there is no current RequestStream the response "(exception, requeststream_not_current)" is returned.

Output parameters

USERID
The userid of the request processor to be joined.

TRANID
The transid of the request processor to be joined.

PUBLIC_ID
The public id of the target RequestStream on the processor to be joined.

REQUEST_DATA_LENGTH
The data length of the request to be passed to the processor to be joined.

RESPONSE
Values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	REQUESTSTREAM_NOT_CURRENT

RZTA gate, TERMINATE function

Terminate the current (target) RequestStream either normally or abnormally. After this call the "XM" token in the transaction instance is cleared and no longer denotes a RequestStream.

If no "rs_token" is specified and there is no current RequestStream ("XM" token) then the response "(exception, RequestStream_not_current)" is returned.

If "rs_token" is specified it is taken as the token of the RequestStream to terminate.

If the RequestStream token is not in the +RegionRSTable+ (or else is not a +target+ RequestStream in that table) then the response "(exception, rs_token_unknown)" is returned.

If the termination is "normal" then transports are tested for being in a state that accepts normal termination and if they are they are closed and storage associated with the RequestStream is returned and the local region token is invalidated. The +ResionRSTable+ has the "rs_token" removed, and the current RequestStream is unset (the "XM" token is cleared but only if the "rs_token" was +not+ specified on the call). If any transports are +not+ in a correct state then no action is taken and the response "(exception, cannot_terminate_normally)" is returned.

If the termination is "abnormal" then the transports are closed in whatever state they are found, and the RequestStream is terminated as for normal termination.

The usual exceptions concerning the transport mechanism (if used) are possible however +the RequestStream was still terminated even if a transport exception is returned+.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	REQUESTSTREAM_NOT_CURRENT RS_TOKEN_UNKNOWN SERVICE_NOT_AVAILABLE TRANSPORT_FAILURE CANNOT_TERMINATE_NORMALLY

RZRT gate, SET_EXIT_PROGRAM function

The following defines the syntax of the SET_EXIT_PROGRAM function.

This is used to identify the name of the distributed routing user-replaceable program at domain initialization time and when the program name is dynamically changed. During CICS initialization the local sysid is also passed to "RZ" from Parameter Manager via this interface.

Input parameters

PROGRAM_NAME The name of the user-replaceable program for the Distributed Dynamic Routing program.

[LOCAL_SYSID]
The SYSID for the local CICS region to recognize it in routing user-replaceable program responses.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

RZRJ gate, PERFORM_JOIN function

This function reduces the calls necessary from the join task (in remote join capability) to the RZ domain. It initiates the procedures necessary to pass an attached RequestStream to a local processor.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

	`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`
REASON	Returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TRANSPORT_FAILURE JOIN_NOT_POSSIBLE

RequestStream domain's generic gates

Table 90 summarizes the generic gates which the RequestStream domain implements. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 90. RequestStream domain's generic gate calls

Gate	Trace	Function	Format
RZDM	RZ 0101 RZ 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

For descriptions of these functions and their input and output parameters, refer to the Sections dealing with the corresponding generic format:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669

RequestStream domain formats

Table 91 shows the formats owned by the requeststream domain, and shows the functions performed. The modules that issue requests in this format make them to RZ.

Table 91. Generic formats owned by the RequestStream domain.

Format	Module called	Calling modules	Functions
LSTN	DFHRZLN	DFHIIDM DFHIIRP DFHIIRR	REGISTER LISTEN CANCEL DEREGISTER
NOTI	DFHIIRP DFHIIRR	DFHRZLN	NOTIFY

LSTN gate, REGISTER function

Register the notify interface with the server domain. The calback_gate is the gate number +in the caller's domain+. The caller's domain is inferred by kernel linkage.

Returns the "notify_token" which identifies this registration.

Returns ("exception, registration_rejected") if the domain is not ready to allow registrations (initializing or quiescing, for example), or if it cannot issue any more tokens (due to resource limitations, for example). This response is given if the other functions of the domain are not compromised by this failure. Higher severity responses may be given otherwise.

The caller must supply the parameters "call_domain()" and "call_gate()" on this function. See design file for details.

Input parameters

CALLBACK_GATE
The gate number of the Domain Gate to be called for notification.

Output parameters

NOTIFY_TOKEN The token identifying this notification registration. Used on Listen, Cancel, and Notify functions.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	REGISTRATION_REJECTED

LSTN gate, LISTEN function

Listen to the server object and notify the registered notify callback gate, identified by "notify_token", if the object requires service. Pass the supplied "client_token" on the notify call.

Exceptions:

"notify_token_unknown"
the notify token cannot be found by this server domain;

"notify_token_in_use"
the notify token is being used and this server does not allow multiple uses;

"notify_token_misused"
the notify token belongs to another domain and this server does not allow multiple client access to it;

"server_token_unknown"
the server does not recognize the server token;

"server_token_in_use"
the object denoted by the server token already has a listen outstanding, and this server does not allow multiple listens.

The caller must supply the parameters "call_domain()" and "call_gate()" on this function. See design file for details.

Input parameters

NOTIFY_TOKEN Token that identifies the registration for this listen request.

SERVER_TOKEN Token that identifies the server-owned object being listened to.

CLIENT_TOKEN Token for the client, returned on Notify to identify this listen request.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOTIFY_TOKEN_UNKNOWN NOTIFY_TOKEN_IN_USE NOTIFY_TOKEN_MISUSED SERVER_TOKEN_UNKNOWN SERVER_TOKEN_IN_USE

LSTN gate, CANCEL function

Cancel an outstanding listen request. Ensure that it is not driven upon return from this call. The "notify_token" and the "server_token" are to be specified, and optionally the "client_token" that was specified on the "listen" request can be retrieved.

Exceptions:

"notify_token_unknown"
: the notify token cannot be found by this server domain;

"notify_token_misused"
: the notify token belongs to another domain and this server does not allow multiple client access to it;

"server_token_unknown"
: the server does not recognize the server token;

"listen_not_outstanding"
: the object denoted by the server token does not have a listen outstanding.

The caller must supply the parameters "call_domain()" and "call_gate()" on this function. See design file for details.

Input parameters

NOTIFY_TOKEN Token that identifies the registration for this cancel request.

SERVER_TOKEN Token that identifies the server-owned object being listened to.

Output parameters

[CLIENT_TOKEN]
: Client Token that was passed on LISTEN and would have been passed to NOTIFY.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOTIFY_TOKEN_UNKNOWN NOTIFY_TOKEN_MISUSED SERVER_TOKEN_UNKNOWN LISTEN_NOT_OUTSTANDING

LSTN gate, DEREGISTER function

Call outstanding notifies on this registration (identified by "notify_token") with a normal termination notification, and then remove the registration of the client (caller) domain from this server domain. The "notify_token" is no longer valid.

Exceptions:

"notify_token_unknown"
: the notify token cannot be found by this server domain;

"notify_token_misused"
: the notify token belongs to another domain and this server does not allow multiple client domain access to it;

The caller must supply the parameters "call_domain()" and "call_gate()" on this function. See design file for details.

Input parameters

NOTIFY_TOKEN Token that identifies the registration token being deregistered.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOTIFY_TOKEN_UNKNOWN, NOTIFY_TOKEN_MISUSED

NOTI gate, NOTIFY function

Call the client domain notify gate registered as "notify_token" passing the "server_token" and the "client_token" given on the "listen" call, and with a status indicating the notification reason. The listen is considered discharged after this call, and this notify will not be called again unless another "listen" request is made.

If the callback returns an exception response then the registration should be deleted. This is equivalent to a "deregister" call, including the call of this same notify callback gate for any other outstanding "listen" requests.

If the callback returns a response more serious than exception ("disaster", "purged", etc.) then the registration should be deleted but no other callbacks are to be made to outstanding "listen"s for this registration. In particular a kernel error, or an "invalid" response, should not attempt to recall the same gate again. This is deemed to be a severe internal error.

The caller (the server domain) will use "call_gate()" with the value of the callback gate supplied on registration, and "call_domain()" with the client domain who registered. This must refer to a valid gate or a kernel error will result.

Input parameters

NOTIFY_TOKEN Token that identifies the registration for this notify request.

SERVER_TOKEN Token that identifies the server-owned object being notified.

CLIENT_TOKEN Token supplied by the client on the listen request.

NOTIFY_STATUS

Values: **NOTIFY, CLOSE, ABEND, TIMEOUT**

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON Returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOTIFY_CALLBACK_FAILED

Modules

The following modules implement the RZ domain:

Module	Function
DFHRZDUF	Dump Formatting program
DFHRZIX	XM Attach Client for InStore transports
DFHRZJN	Join task program
DFHRZLN	Listen and Notify calls
DFHRZNR2	Init rsnr class (notification object)
DFHRZOFI	Init object factory class
DFHRZRG2	Init rsrg registration class

Module	Function
DFHRZRJ	Perform join
DFHRZRM	RM Resource Owner for RZ
DFHRZRS1	Init rz_reqstream class
DFHRZRT	Set Routing Exit program name
DFHRZRT1	Init routing user-replaceable program class (rzrt)
DFHRZRT2	Invoke Routing user-replaceable program
DFHRZSO	Source commands on RequestStreams (not Create/Join)
DFHRZSO1	Create and Join commands on Source RequestStreams
DFHRZTA	Target commands on RequestStreams
DFHRZTCX	XM Attach Client for MRO transports
DFHRZTRI	Trace interpretation
DFHRZTR1	Init rztr class
DFHRZVP1	Init rzvp class
DFHRZXM	XM Attach Client for RequestStreams

Exits

None

User-replaceable programs

Program DFHRZRT2 calls the dynamic routing program, which is described in *CICS Customization Guide*.

Trace

The point IDs for the RequestStream domain are of the form RZ xxxx; the corresponding trace levels are RZ 1, RZ 2 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 102. Scheduler Services domain (SH)

The scheduler services domain (also sometimes known simply as "scheduler services") is used to harden schedule requests between UOWs and to route schedule requests to a target region identified by the distributed routing exit program. A schedule request may be viewed as a request to undertake a piece of work, execute a named transaction. The domain is part of CICS business transaction services.

Scheduler services domain's specific gate

Table 92 summarizes the scheduler services domain's specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 92. Scheduler services domain's specific gate

Gate	Trace	Function	XPI
SHPR	SH 0151 SH 0152	ADD_PENDING_REQUEST DELETE_PENDING_REQUEST SET_BOUND_REQUEST	NO NO NO
SHRT	SH 0141 SH 0142	SET_EXIT_PROGRAM INQUIRE_EXIT_PROGRAM	NO NO
SHRQ	SH 0111 SH 0112	PERFORM_RESTART_DREDGE PERFORM_REGULAR_DREDGE PERFORM_SHUTDOWN	NO NO NO
SHRR	SH 0161 SH 0162	ROUTE_REQUEST RECEIVE_REQUEST RETRY_REQUEST	NO NO NO

SHPR gate, ADD_PENDING_REQUEST function

The ADD_PENDING_REQUEST function of the SHPR gate is used to add a pending schedule request to the scheduler services queue associated with this UOW. The pending schedule requests are hardened to the scheduler services local request queue (LRQ) as part of syncpoint processing.

Input parameters

TRANID is an 4-character transaction id.

USERID is an 8-character userid.

TIME is a string of length 8, used when a request is delayed for a period time.

TOKEN is a string of length 4, used to identify the pending queue.

BALANCE indicates whether this schedule request is eligible for workload balancing. It can have either of these values:

YES|NO

PTYPE is the 8-character process type.

PNAME is the 36-character process name.

ACTIVITY_ID is a block containing the activity id.

ACTIVITY_REQUEST_BLOCK
is a block containing the BAM domain activity request block.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

SHPR gate, DELETE_PENDING_REQUEST function

The DELETE_PENDING_REQUEST of the SHPR gate is used to delete a pending request queue.

Input parameters

TOKEN is a string of length 4, which identifies the queue to be deleted.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|PURGED|INVALID|DISASTER|KERNERROR

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	REQUEST_NOT_FOUND

SHPR gate, SET_BOUND_REQUEST function

The SET_BOUND_REQUEST function of the SHPR gate is used to update the schedule request to indicate that a process and/or activity has completed.

Input parameters

ACTIVITY_COMPLETE
indicates whether the activity associated with this UOW has completed. It can have either of these values:

YES|NO

PROCESS_COMPLETE
indicates whether the process associated with this UOW has completed. It can have either of these values:

YES|NO

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	REQUEST_NOT_FOUND

SHRT gate, SET_EXIT_PROGRAM function

The SET_EXIT_PROGRAM function of the SHRT gate is used to alter the distributed routing exit program, initially named on the DSRTPGM system initialisation parameter. The sysid of the local system is passed during CICS initialisation.

Input parameters

PROGRAM_NAME is the 8-character exit program name.

LOCAL_SYSID is the 4-character local sysid.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

SHRT gate, INQUIRE_EXIT_PROGRAM function

The INQUIRE_EXIT_PROGRAM function of the SHRT gate is used to return the name of the distributed routing exit program, initially named on the DSRTPGM system initialisation parameter.

Input parameters

PROGRAM_NAME is the 8-character exit program name.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND

SHRQ gate, PERFORM_RESTART_DREDGE function

The PERFORM_RESTART_DREDGE of the SHRQ gate is used to initiate the dredging of expired schedule requests on the local request queue (LRQ) after a CICS system restart.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

SHRQ gate, PERFORM_REGULAR_DREDGE function

The PERFORM_REGULAR_DREDGE function of the SHRQ gate initiates the periodic dredging of expired schedule requests on the local request queue (LRQ).

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

SHRQ gate, PERFORM_SHUTDOWN function

The PERFORM_SHUTDOWN function of the SHRQ gate is used to stop dredging of schedule requests on the local request queue (LRQ), preventing any further CICS BTS work from being initiated.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

SHRR gate, ROUTE_REQUEST function

The ROUTE_REQUEST function of the SHRR gate is used to identify a target region to which a schedule request should be routed.

Input parameters

REQUEST_BUFFER
is a buffer used to hold the schedule request which is to be routed.

Output parameters

SYSID is the 4-character sysid of the region to which the schedule request should be routed.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_REQUEST_FOUND REQUEST_BUFFER_TOO_SMALL NO_SYSTEM

SHRR gate, RECEIVE_REQUEST function

The RECIEVE_REQUEST function of the SHRR gate is used to receive a schedule request once it has been routed to the target region.

Input parameters

REQUEST_BUFFER
is a buffer used to hold the received schedule request.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_REQUEST_RECEIVED

SHRR gate, RETRY_REQUEST function

The RETRY_REQUEST function of the SHRR gate is used obtain another target region if the initial attempt at routing the schedule request fails.

Input parameters

REQUEST_BUFFER
is a buffer used to hold the schedule request which is to be routed.

ROUTE_ERROR
indicates the reason why the routing of the schedule request failed. It can have a value of:

SYSID_NOT_FOUND|SYSID_OUT_OF_SERVICE|NO_SESSIONS|
ALLOCATE_REJECTED|QUEUE_PURGED|FUNC_NOT_SUPPORTED|
LEGERR|PGMIDERR|INVREQ|NOTAUTH|TERMERR

Output parameters

SYSID
is the 4-character sysid of the region to which the schedule request should be routed.

LOCAL
indicates whether we should retry the schedule request on the local region. It can take the values:

YES|NO

ABEND_CODE
is the 4-character abend code.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_REQUEST_FOUND REQUEST_BUFFER_TOO_SMALL NO_SYSTEM

Scheduler service domain's generic gates

Table 93 summarizes the scheduler services domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 93. Scheduler services domain's generic gates

Gate	Trace	Function	Format
DMDM	SH 0101 SH 0102	PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
XMAC	SH 0121 SH 0122	INIT_XM_CLIENT BIND_XM_CLIENT RELEASE_XM_CLIENT	XMAC
RMDE	SH 0131 SH 0132	START_DELIVERY DELIVER_RECOVERY END_DELIVERY	RMDE
RMKP	SH 0131 SH 0132	TAKE_KEYPOINT	RMKP
RMRO	SH 0131 SH 0132	PERFORM_PERPARE PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT START_BACKOUT DELIVER_BACKOUT END_BACKOUT	RMDE
TISR	SH 0701 SH 0702	NOTIFY	TISR
KETI	SH 0701 SH 0702	NOTIFY_RESET	KETI

For descriptions of these functions and their input and output parameters, refer to the Sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—“Domain manager domain's generic formats” on page 669
Format XMAC—Chapter 113, “Transaction manager domain (XM),” on page 1275
Format RMDE—“Recovery Manager domain's call back formats” on page 1088
Format RMRO—“Recovery Manager domain's call back formats” on page 1088
Format RMKP—“Recovery Manager domain's call back formats” on page 1088
Format TISR—“Timer domain's specific gate” on page 1201
Format KETI—“Kernel domain's specific gates” on page 831

When invoked for the DMDM INITIALIZE_DOMAIN function scheduler services obtains its anchor block and initializes its various classes. This would include starting the scheduler services system task , CSHY and obtaining the name of the distributed routing exit program named on the DSRTPGM system initialization parameter.

When invoked by transaction manager via the XMAC generic gate, for INIT_XM_CLIENT SH domain obtains a user token in order to set up the correct transaction environment. For BIND_XM_CLIENT SH domain initializes recoverable resources, which includes setting the RM work token and logging a backout request for this UOW. SH domain also determines the name of the program to be invoked on the initial program link.

When invoked for the RMRO PERFORM_PREPARE function SH domain prepares to commit the pending request for the UOW by adding them to the local request queue (LRQ). On receipt of the RMRO PERFORM_COMMIT the schedule requests for this UOW are committed or destroyed, depending upon whether we are committing forwards or backwards.

When invoked for the RMDE DELIVER_RECOVERY function SH domain recreates the pending request queues and in the case of inflight UOWs attempts to retry the associated BTS activation.

Scheduler services makes use of the TISR functions, REQUEST_ NOTIFY_INTERVAL and NOTIFY to deal with delayed schedule requests i.e. EXEC CICS DEFINE TIMER calls.

The KETI interface is used when the time is adjusted, causing the time at which delayed schedule requests are to expire to be recalculated.

Modules

Module	Function
DFHSHDM	Handles the following requests: PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHSHRM	Handles the following requests: PERFORM_PREPARE PERFORM_COMMIT START_BACKOUT DELIVER_BACKOUT END_BACKOUT PERFORM_SHUNT PERFORM_UNSHUNT TAKE_KEYPOINT START_DELIVERY DELIVER_RECOVERY END_DELIVERY
DFHSHXM	Handles the following requests: INIT_XM_CLIENT BIND_XM_CLIENT RELEASE_XM_CLIENT
DFHSHTI	Handles the following requests: NOTIFY NOTIFY_RESET
DFHSHRQ	Handles the following requests: PERFORM_RESTART_DREDGE PERFORM_REGULAR_DREDGE PERFORM_SHUTDOWN
DFHSHPR	Handles the following requests: ADD_PENDING_REQUEST DELETE_PENDING_REQUEST SET_BOUND_REQUEST
DFHSHRT	Handles the following requests: SET_EXIT_PROGRAM INQUIRE_EXIT_PROGRAM

Module	Function
DFHSHRR	Handles the following requests: ROUTE_REQUEST RECEIVE_REUEST RETRY_REQUEST
DFHSHSY	Implements the SH domain system task, CSHY.
DFHSHRRP	The SH domain request receiving program, the back-end to SH domain DPL requests.
DFHSHRSP	The SH domain request sending program, the front-end to SH domain DPL requests.
DFHSHDUF	Formats the SH domain control blocks
DFHSHTRI	Interprets SH domain trace entries
DFHSHRE1	Initializes the SH domain request class.
DFHSHOFI	Initializes the SH domain object factory class.
DFHSHVP1	Initializes the SH domain variable length storage class.
DFHSHRT1	Initializes the SH domain request routing class.
DFHSHRT2	Invokes the distributed routing exit program, named on the DSRTPGM system initialization parameter.
DFHSHRQ1	Initializes the SH domain request queue class.

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the scheduler services domain are of the form SH xxxx; the corresponding trace levels are SH 1, SH 2, and Exc.

For more information about the trace points, see *CICS Trace Entries*. For more information about using traces in problem determination, see *CICS Problem Determination Guide*.

Chapter 103. JVM domain (SJ)

The JVM domain provides services to:
- Invoke a Java program
- Inquire on and set the JVM pool
- Inquire on individual JVMs
- Inquire on JVM profiles and set the directory for them
- Manage the shared class cache
- Delete inactive JVMs

This Section covers:

SJ domain: Design overview

The process by which CICS creates JVMs and allocates them to requests is described in “About JVMs” in *Java Applications in CICS*. This design overview looks at that process from a more technical viewpoint.

The actions that CICS takes when a Java program is invoked are as follows:
1. When the Java program is started, the Program Manager (PG) domain recognizes from the PROGRAM resource definition that it is a Java program. It calls the APLJ gate's START_PROGRAM function, which calls the SJIN gate's INVOKE_JAVA_PROGRAM function in the JVM domain.
2. The INVOKE_JAVA_PROGRAM function determines if the program is to run in CICS key or in user key (from the EXEC_KEY input parameter), and calls the Dispatcher (DS) domain's CHANGE_MODE function to move the task to either a J8 TCB (for CICS key) or a J9 TCB (for user key).
3. The Dispatcher domain looks for an existing J8 or J9 TCB, in the pool of JVM TCBs (the JVM pool), that matches the execution key and the JVM profile (JVM_PROFILE_NAME input parameter) requested by the program. “How CICS allocates JVMs to applications” in *Java Applications in CICS* explains the selection mechanism that the Dispatcher domain uses to decide whether to assign the request an existing, matching TCB, or to assign the request a new TCB, or to destroy and re-create an existing, mismatching TCB, or to make the request wait. If the Dispatcher domain assigns the request a new TCB or a re-created TCB, steps 4 and 5 of this process must now be performed. If the Dispatcher domain assigns the program request an existing, matching TCB, steps 4 and 5 are omitted, and the request re-enters the process at step 6.
4. If the Dispatcher domain assigns the request a new TCB or a re-created TCB, the JVM (SJ) domain must build a JVM on the TCB. To do this, it obtains storage in the appropriate key from the Storage Manager (SM) domain to keep its representation of the TCB. The address of this storage is used as a token, and returned to the Dispatcher domain for it to track. The SJ domain calls Language Environment using the Language Environment preinitialization module CEEPIPI (in the Application (AP) domain). CEEPIPI starts a Language Environment enclave on the TCB. “The structure of a JVM” in *Java Applications in CICS* explains more about the relationship between the Language Environment enclave and the JVM.
5. The SJCS program (a CICS program written in C) runs in the Language Environment enclave. It processes the JVM profile (JVM_PROFILE_NAME input parameter) and the associated JVM properties

file for the request, to build a JVM with the appropriate attributes. “How CICS creates JVMs” in *Java Applications in CICS* explains what attributes of a JVM can be specified in its JVM profile and JVM properties file. If the JVM profile indicates that this JVM is to use the shared class cache (that is, it is a worker JVM), a call is made to the shared class cache to obtain the token for the currently active JVMset. “The shared class cache” in *Java Applications in CICS* explains more about the relationship between worker JVMs and the shared class cache. The JNI function JNI_StartJavaVM is then used to start the JVM, and this call returns two JVM tokens, which are stored in a SJ control block for later use. The JVM has now been created, and the SJCS program returns to SJIN.

6. If the Dispatcher domain assigned the program request an existing, matching TCB, which has a JVM that has already been created, the request re-enters the process at this point. The SJ domain now calls another C subroutine to invoke the Wrapper class. The wrapper loads required DLLs (to support CICS' native methods) and output redirection classes, then it calls our user class (the Java program), as specified by the USER_CLASS input parameter.
7. When the user class (the Java program) returns, if the option REUSE=RESET was specified in the JVM profile, creating a resettable JVM, a JVM reset is attempted. “How JVMs can be reset for reuse” in *Java Applications in CICS* has more information about the conditions that must be met for a JVM to be reset, and why sometimes they cannot be reset. If the JVM reset is attempted but fails, the JVM and the Language Environment enclave are destroyed, but the TCB is not destroyed (unless the JVM failed with an abend). If the JVM reset succeeds, the JVM and TCB are ready for reuse, and the Dispatcher domain can assign them to another program request (as described in step 3). If the option REUSE=YES was specified in the JVM profile, creating a continuous JVM, a JVM reset is not attempted, and the JVM and TCB are ready for reuse as soon as the user class returns (unless garbage collection is taking place in the JVM). If the option REUSE=NO was specified in the JVM profile, creating a single-use JVM, the JVM is not made available for reuse, but instead the JVM and the Language Environment enclave are destroyed as soon as the user class returns.

The other functions provided by the JVM domain are to do with the management of the JVM pool (the pool of J8 and J9 TCBs that are used to build JVMs to service Java program requests), and of the shared class cache. “How CICS manages JVMs in the JVM pool” in *Java Applications in CICS* explains the part CICS plays in this process, and “Managing the shared class cache” and “Managing your JVMs” in *Java Applications in CICS* describe the functions that system administrators can use.

SJ domain’s specific gates

Table 94 summarizes the SJ (JVM) domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 94. SJ domain’s specific gates

Gate	Trace	Function	XPI
SJCC	SJ 0601	START_CLASSCACHE	NO
	SJ 0602	STOP_CLASSCACHE	NO
		RELOAD_CLASSCACHE	NO
		NOTIFY_CLASSCACHE	NO
		ADD_TO_ACTIVE_JVMSET	NO
		REMOVE_FROM_JVMSET	NO
SJIN	SJ 0201	INVOKE_JAVA_PROGRAM	NO
	SJ 0202	(also has a generic function: NOTIFY_DELETE_TCB)	

Table 94. SJ domain's specific gates (continued)

Gate	Trace	Function	XPI
SJIS	SJ 0301	INQUIRE_JVMPOOL	NO
	SJ 0302	SET_JVMPOOL	NO
		INQUIRE_JVM	NO
		START_BROWSE_JVM	NO
		GET_NEXT_JVM	NO
		END_BROWSE_JVM	NO
		INQUIRE_CLASSCACHE	NO
		SET_CLASSCACHE	NO
		INQUIRE_JVMPROFILE	NO
		START_BROWSE_JVMPROFILE	NO
		GET_NEXT_JVMPROFILE	NO
		END_BROWSE_JVMPROFILE	NO
		SET_JVMPROFILEDIR	NO
		DELETE_INACTIVE_JVMS	NO

SJCC gate, START_CLASSCACHE function

The START_CLASSCACHE function of the SJCC gate is used to start the shared class cache.

Input parameters

[CACHE_SIZE] The size of the shared class cache.

[JVM_PROFILE_NAME]
The name of the JVM profile to be used for the master JVM that initializes the shared class cache.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_STOPPED

SJCC gate, STOP_CLASSCACHE function

The STOP_CLASSCACHE function of the SJCC gate is used to stop the shared class cache.

Input parameters

[AUTOSTART] The autostart status that is to be set for the shared class cache, to determine whether or not it will restart automatically when a JVM requests its use. It can have the values:

ENABLED|DISABLED

[TERMINATE] The type of termination that is to be attempted for the shared class cache and the worker JVMs that are dependent on it. It can have the values:

PHASEOUT|PURGE|FORCEPURGE

When PHASEOUT is specified, the supporting TCBs for the JVMs will be marked for deletion at the termination of their current task (if any). If PURGE or FORCEPURGE is specified, then premature termination of those tasks is initiated. When all worker JVMs have been terminated, the shared class cache is also terminated.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ALREADY_STOPPED

SJCC gate, RELOAD_CLASSCACHE function

The RELOAD_CLASSCACHE function of the SJCC gate is used to reload the shared class cache—that is, to start a new master JVM and phase out the existing JVMset.

Input parameters

[CACHE_SIZE] The size of the shared class cache.

[JVM_PROFILE_NAME]
The name of the JVM profile to be used for the master JVM that initializes the shared class cache.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_STARTED

SJCC gate, NOTIFY_CLASSCACHE function

The NOTIFY_CLASSCACHE function of the SJCC gate is used to confirm the status of the master JVM that initializes the shared class cache. If the master JVM starts successfully, the function quiesces the previous shared class cache so that the new master JVM becomes the active shared class cache, and can be used by new worker JVMs (so the new master and worker JVMs become the active JVMset). If the master JVM fails to start, its autostart status is marked as disabled.

Input parameters

SJVMS_TOKEN The token of the SJVMS control block.

JVMSET_STATUS
The status of the JVMset, that is, whether the master JVM started successfully. It can have the values:
READY|START_FAILED

[JVMSET_TOKEN]
The token of the master JVM.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	MASTER_JVM_ABENDED
INVALID	INVALID_CC_STATE

SJCC gate, ADD_TO_ACTIVE_JVMSET function

The ADD_TO_ACTIVE_JVMSET function of the SJCC gate is used to add a new worker JVM to the active JVMset, and also to automatically start the shared class cache if autostart is enabled and the shared class cache is not started. The active JVMset consists of the master JVM for the active shared class cache, and the worker JVMs that are dependent on it.

Input parameters

SJTCB_TOKEN The token of the TCB on which the worker JVM is to be built.

Output parameters

JVMSET_TOKEN The token of the master JVM.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	AUTOSTART_DISABLED
DISASTER	INVALID_CC_STATE

SJCC gate, REMOVE_FROM_JVMSET function

The REMOVE_FROM_JVMSET function of the SJCC gate is used to dissociate a worker JVM from the master JVM as part of the termination process for the worker JVM.

Input parameters

SJTCB_TOKEN The token of the TCB on which the worker JVM has been built.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None.

SJIN gate, INVOKE_JAVA_PROGRAM function

The INVOKE_JAVA_PROGRAM function of the SJIN gate is used to invoke a user Java program.

Input parameters

PROGRAM The program name of the program to be invoked.

TRANSACTION The transaction id of the current transaction.

JVM_PROFILE_NAME
The name of the JVM profile to be used for the JVM for this program to run in.

USER_CLASS The name of the main class in the Java program that is to run in the JVM.

EXEC_KEY The EXEC key of the JVM. It can have the values:

CICS|USER

Output parameters

ABEND_CODE The CICS abend code returned if an abend occurs.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	AUTOSTART_DISABLED JVM_START_FAILURE JVM_POOL_DISABLED JVM_PROFILE_MISSING JVM_PROFILE_INVALID SYSTEM_PROPERTIES_MISSING SYSTEM_PROPERTIES_INVALID TRANSACTION_ABENDED USER_CLASS_NOT_FOUND

SJIS gate, INQUIRE_JVMPOOL function

The INQUIRE_JVMPOOL function of the SJIS gate is used to retrieve information about the JVM pool.

Input parameters

[JVM_LEVEL0_TRACE_BUFFER]
is a buffer which is used by the JVM domain to return the JVM trace options that have been set for JVM Level 0 trace (up to 240 characters).

[JVM_LEVEL1_TRACE_BUFFER]
is a buffer which is used by the JVM domain to return the JVM trace options that have been set for JVM Level 1 trace (up to 240 characters).

[JVM_LEVEL2_TRACE_BUFFER]
is a buffer which is used by the JVM domain to return the JVM trace options that have been set for JVM Level 2 trace (up to 240 characters).

[JVM_USER_TRACE_BUFFER]
is a buffer which is used by the JVM domain to return the JVM trace options that have been set for JVM User trace (up to 240 characters).

Output parameters

[TOTAL] The total number of JVMs in the JVM pool.

[STATUS] The status of the JVM pool (that is, whether it can service new requests or not). It can have the values:

ENABLED|DISABLED

[PHASINGOUT] The number of JVMs that are curently being phased out (that is, they have been marked for deletion, but are still being used by a task).

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	JVM_LEVEL0_TRACE_OVERFLOW JVM_LEVEL1_TRACE_OVERFLOW JVM_LEVEL2_TRACE_OVERFLOW JVM_USER_TRACE_OVERFLOW
DISASTER	INSUFFICIENT_STORAGE

SJIS gate, SET_JVMPOOL function

The SET_JVMPOOL function of the SJIS gate is used to set the status of the JVM pool, or to set JVM trace options for the JVM pool, or to terminate the JVM pool.

Input parameters

[STATUS] The status of the JVM pool (that is, whether it can service new requests or not). It can have the values:

ENABLED|DISABLED

[TERMINATE] The type of termination that is to be attempted for the JVMs in the JVM pool (both worker JVMs and standalone JVMs), and for the shared class cache. It can have the values:

PHASEOUT|PURGE|FORCEPURGE

When PHASEOUT is specified, the supporting TCBs for the JVMs will be marked for deletion at the termination of their current task (if any). If PURGE or FORCEPURGE is specified, then premature termination of those tasks is initiated. If a shared class cache has been started, it will be terminated when all the worker JVMs that were dependent on it have been terminated.

[JVM_LEVEL0_TRACE_BLOCK]
is a buffer containing the JVM trace options (up to 240 characters) that are to be set for JVM Level 0 trace.

[JVM_LEVEL1_TRACE_BLOCK]
is a buffer containing the JVM trace options (up to 240 characters) that are to be set for JVM Level 1 trace.

[JVM_LEVEL2_TRACE_BLOCK]
is a buffer containing the JVM trace options (up to 240 characters) that are to be set for JVM Level 2 trace.

[JVM_USER_TRACE_BLOCK]
is a buffer containing the JVM trace options (up to 240 characters) that are to be set for JVM User trace.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None.

SJIS gate, INQUIRE_JVM function

The INQUIRE_JVM function of the SJIS gate is used to identify and retrieve information about the JVMs in the JVM pool.

Input parameters

JVM_ID
The JVM token, a value that identifies the JVM.

Output parameters

[AGE]
The number of seconds since the JVM was initialized.

[ALLOC_AGE]
The number of seconds for which the JVM has been allocated to its task (zero if the JVM is not currently allocated to a task).

[CLASSCACHE]
Indicates whether the JVM is a worker JVM dependent on the shared class cache. It can have the values:

YES|NO

[EXEC_KEY]
The EXEC key of the JVM. It can have the values:

CICS|USER

[PHASING_OUT]
Indicates whether the JVM is being phased out (that is, it has been marked for deletion, but is still being used by a task). It can have the values:

YES|NO

[JVMPROFILE_NAME]
The name of the JVM profile used to initialize the JVM.

[TRANNUM]
The task to which the JVM is allocated (zero if the JVM is not currently allocated to a task).

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	JVM_NOT_FOUND

SJIS gate, START_BROWSE_JVM function

The START_BROWSE_JVM function of the SJIS gate starts a browse of the JVMs in the JVM pool.

Input parameters

None.

Output parameters

BROWSE_TOKEN
A pointer to the JVM_ID (JVM token) of the first JVM that is to be browsed.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None.

SJIS gate, GET_NEXT_JVM function

The GET_NEXT_JVM function of the SJIS gate returns the next JVM in the JVM pool. The JVMs are ordered by their JVM tokens.

Input parameters

BROWSE_TOKEN A pointer to the JVM_ID (JVM token) of the last JVM that was found by the browse.

Output parameters

[JVM_ID] The JVM token, a value that identifies the JVM.

[AGE] The number of seconds since the JVM was initialized.

[ALLOC_AGE] The number of seconds for which the JVM has been allocated to its task (zero if the JVM is not currently allocated to a task).

[CLASSCACHE] Indicates whether the JVM is a worker JVM dependent on the shared class cache. It can have the values:

YES|NO

[EXEC_KEY] The EXEC key of the JVM. It can have the values:

CICS|USER

[PHASING_OUT]
Indicates whether the JVM is being phased out (that is, it has been marked for deletion, but is still being used by a task). It can have the values:

YES|NO

[JVMPROFILE_NAME]
The name of the JVM profile used to initialize the JVM.

[TRANNUM] The task to which the JVM is allocated (zero if the JVM is not currently allocated to a task).

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	END_OF_BROWSE

SJIS gate, END_BROWSE_JVM function

The END_BROWSE_JVM function of the SJIS gate ends the browse of the JVMs in the JVM pool.

Input parameters

BROWSE_TOKEN A pointer to the JVM_ID (JVM token) of the last JVM that was found by the browse.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None.

SJIS gate, INQUIRE_CLASSCACHE function

The INQUIRE_CLASSCACHE function of the SJIS gate is used to retrieve information about the shared class cache in the CICS region.

Input parameters

None.

Output parameters

[ACTIVE_JVMS]
The number of worker JVMs in the CICS region that are dependent on the current shared class cache or on a shared class cache that is phasing out.

[AUTOSTART] The status of autostart for the shared class cache.

[JVMPROFILE_NAME]
The name of the JVM profile for the master JVM that initializes the shared class cache.

[PHASINGOUT_JVMS]
The number of worker JVMs that are dependent on an old shared class cache (or on the current shared class cache, if its status is STOPPED) and are being phased out.

[PHASINGOUT_JVMSETS]
The number of old shared class caches that are still present in the region because they are waiting for worker JVMs that are dependent on them to be phased out (including the current shared class cache, if its status is STOPPED).

[CACHE_FREE] The amount of free space in the shared class cache.

[CACHE_SIZE] The size of the shared class cache, in bytes.

[START_DATE] The date on which the current shared class cache was started.

[START_TIME] The time at which the current shared class cache was started.

[START_ABSTIME]
The absolute date and time at which the current shared class cache was started (ABSTIME format).

[STARTED_STATUS]
The status of the current shared class cache (STARTING, STARTED, RELOADING or STOPPED).

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None.

SJIS gate, SET_CLASSCACHE function

The SET_CLASSCACHE function of the SJIS gate is used to set the autostart status and size of the shared class cache, and the JVM profile that is to be used for the master JVM.

Input parameters

[AUTOSTART] The autostart status that is to be set for the shared class cache, to determine whether or not it will restart automatically when a JVM requests its use. It can have the values:

ENABLED|DISABLED

[INITIAL_START]
Specifies whether or not the shared class cache will start automatically at CICS initialization. It can have the values:

YES|NO

[JVMPROFILE_NAME]
The name of the JVM profile for the master JVM that initializes the shared class cache.

[CACHE_SIZE] The size of the shared class cache, in bytes.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None.

SJIS gate, INQUIRE_JVMPROFILE function

The INQUIRE_JVMPROFILE function of the SJIS gate is used to retrieve information about JVM profiles that have been used during the lifetime of this CICS region.

Input parameters

JVMPROFILE_NAME
The name of the JVM profile.

JVMPROFILE_PATH_NAME
is a buffer which is used by the JVM domain to return the full path name of the HFS file for the JVM profile (up to 240 characters).

Output parameters

CLASSCACHE
Indicates whether JVMs that use this JVM profile are worker JVMs dependent on the shared class cache. It can have the values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	JVMPROFILE_NOT_FOUND

SJIS gate, START_BROWSE_JVMPROFILE function

The START_BROWSE_JVMPROFILE function of the SJIS gate starts a browse of the JVM profiles that have been used during the lifetime of this CICS region.

Input parameters

None.

Output parameters

BROWSE_TOKEN
A pointer to the first JVM profile to be browsed.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None.

SJIS gate, GET_NEXT_JVMPROFILE function

The GET_NEXT_JVMPROFILE function of the SJIS gate returns the next JVM profile. The JVM profiles are returned in alphabetical order.

Input parameters

BROWSE_TOKEN
A pointer to the last JVM profile that was found by the browse.

JVMPROFILE_PATH_NAME
is a buffer which is used by the JVM domain to return the full path name of the HFS file for the JVM profile (up to 240 characters).

Output parameters

JVMPROFILE_NAME
The name of the JVM profile.

CLASSCACHE
Indicates whether JVMs that use this JVM profile are worker JVMs dependent on the shared class cache. It can have the values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	END_OF_BROWSE
DISASTER	INVALID_BROWSE_TOKEN

SJIS gate, END_BROWSE_JVMPROFILE function

The END_BROWSE_JVMPROFILE function of the SJIS gate ends the browse of the JVM profiles.

Input parameters

BROWSE_TOKEN
A pointer to the last JVM profile that was found by the browse.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None.

SJIS gate, SET_JVMPROFILEDIR function

The SET_JVMPROFILEDIR function of the SJIS gate is used to set the HFS directory where CICS will look for JVM profiles.

Input parameters

JVMPROFILE_DIR_BLOCK
is a buffer containing the full path of the HFS directory where CICS will look for JVM profiles (up to 240 characters).

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None.

SJIS gate, DELETE_INACTIVE_JVMS function

The DELETE_INACTIVE_JVMS function of the SJIS gate is used when MVS storage is constrained, and CICS needs to delete JVMs in the JVM pool that are not currently in use, together with their TCBs.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None.

SJ domain's generic gates

Table 95 summarizes the SJ domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 95. SJ domain's generic gates

Gate	Trace	Function	Format
SJDM	SJ 0000 SJ 0001	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
SJIN	SJ 0201 SJ 0202	NOTIFY_DELETE_TCB	DSAT
SJSM	SJ 0901 SJ 0902	MVS_STORAGE_NOTIFY	SMNT
SJST	SJ 0401 SJ 0402	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

For descriptions of these functions and their input and output parameters, refer to the Sections dealing with the corresponding generic formats:

Functions and parameters

Format DSAT—“Dispatcher domain’s generic formats” on page 717
Format DMDM—Chapter 78, “Domain manager domain (DM),” on page 663
Format SMNT—“Storage manager domain’s generic formats” on page 1159
Format STST—“System programming command flows” on page 264

The Domain Manager gates perform normal internal state initialization and termination functions.

SJ domain: Control blocks

The principal control blocks in the SJ (JVM) domain are as follows:

Control block	Function
SJA	JVM domain anchor block.
SJCCH	Shared class cache control block. Controls all shared class caches (the current, or active, shared class cache; any old shared class caches still in the system; and any new shared class cache that is being loaded).
SJVMS	Master JVM control block (or JVMSet control block). One for each master JVM, the JVM that initializes and owns the shared class cache—so one for each shared class cache. Also known as the JVMSet control block, because JVMSet is another name for the group consisting of a master JVM and shared class cache, and all the worker JVMs that share that class cache.
SJTCB	JVM control block. Tracks the J8 or J9 TCB on which a JVM is built. One for each worker or standalone JVM in the JVM pool (but not master JVMs, which have an SJVMS control block). Each control block ends with a history section that records the task number, transid, and program name of each program invocation against that JVM. The list wraps after 32 entries. It contains an index value, which appears immediately before the History section of the control block, and which indicates the position in the list of the last entry to be added.

“The shared class cache” in *Java Applications in CICS* explains more about the relationship between worker JVMs and the shared class cache.

Figure 111 on page 1139 shows how the control blocks are related. In this example:

- There are six JVMs in the JVM pool, each controlled by an SJTCB control block.
- The first three JVMs are standalone JVMs (that is, they are not using a shared class cache).
- The remaining three JVMs are worker JVMs (that is, they are using a shared class cache). Two worker JVMs are using the current, or active, shared class cache. One worker JVM (JVM 4) is dependent on an old shared class cache, so when the task to which that JVM is allocated is complete, the JVM will be terminated.
- There are three shared class caches in the CICS region, each controlled by an SJVMS control block. Only one shared class cache is active, meaning that new worker JVMs, or worker JVMs starting new tasks, can use it. CICS only supports one active shared class cache in each region. Two JVMs are using (or dependent on) the active shared class cache.
- As well as the active shared class cache, there is an old shared class cache, which is still present in the CICS region because one worker JVM (JVM 4) is dependent on it and has not yet completed its task. As soon as JVM 4 has completed its task and been terminated, the old shared class cache will also be terminated.

- The CICS region also contains a new shared class cache, which is being loaded as a result of a PERFORM CLASSCACHE RELOAD command. (The new shared class cache has been made larger than the active shared class cache.) Worker JVMs cannot yet use this shared class cache. When the new shared class cache is ready, it becomes the active shared class cache, and the previous shared class cache becomes an old shared class cache.

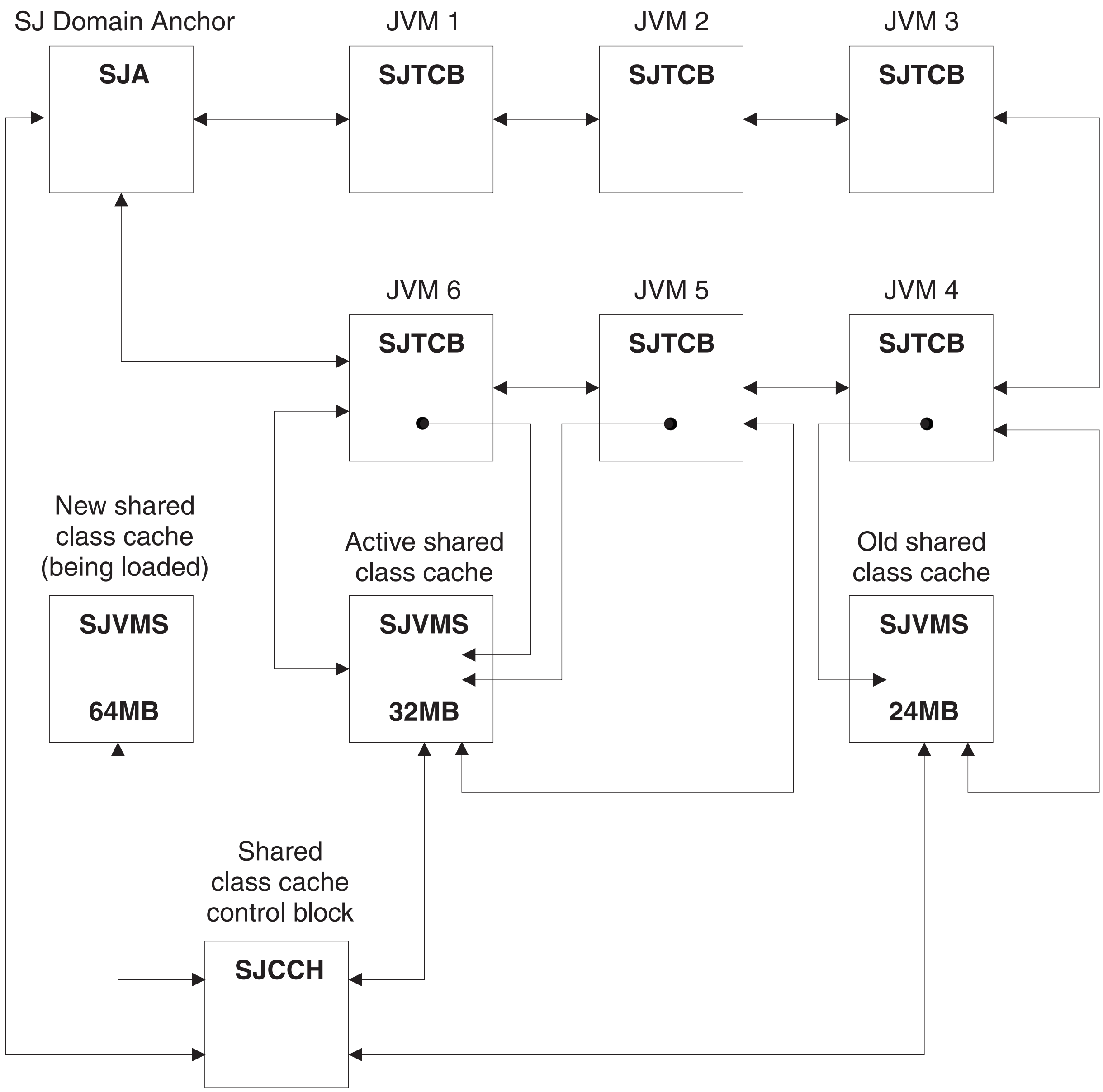


Figure 111. Control blocks associated with the JVM pool and shared class cache

SJ domain: Modules

Module	Function
DFHSJCS	An internal module which handles the following C subroutines called by SJIN: sjcsbld (sjcs_build_jvm) sjcsdes (sjcs_destroy_jvm) sjcscall (sjcs_call_java_method) sjcsrset (sjcs_reset_jvm_output_streams)
DFHSJDM	Handles requests associated with the DMDM generic gate.
DFHSJIN	Handles requests associated with the SJIN gate.
DFHSJIS	Handles requests associated with the SJIS gate.
DFHSJJL	An internal module which handles C subroutines called by the launcher program for the master JVM that initializes the shared class cache.
DFHSJJM	Launcher program for the master JVM that initializes the shared class cache.
DFHSJPJP	An internal module which handles the following C subroutine called by SJIN: sjpjp_process_jvm_profile
DFHSJSM	Handles MVS storage notifications and takes action to reduce the usage of MVS storage if necessary.

SJ domain: Exits

Two user-replaceable programs are used by the SJ domain:
- DFHJVMRO, which is loaded by the SJ domain and used to set user-specified options for an Language Environment enclave in which a JVM is to be started.
- DFHJVMAT, which can be called during the startup of a single-use JVM (one with REUSE=NO or the older option `Xresettable=NO` in its JVM profile), and allows users to interrogate and possibly alter environment variables in order to modify the starting JVM's properties.

For further information about both these user-replaceable programs, see the *CICS Customization Guide*.

You can also customize JVMs by creating Java classes that intercept the stdout and stderr output from a JVM, add time stamps and record headers, and redirect the output to the destination of your choice. “Writing Java classes to redirect JVM stdout and stderr output” in the *CICS Customization Guide* tells you how to do this.

SJ domain: Trace

The point IDs for the SJ domain are of the form SJxxxx. The SJ domain includes trace points relating to the creation and management of JVMs, and to the process of setting JVM trace options. There is also a level 2 trace point SJ 0224, which shows you a history of the programs that have used each JVM. The corresponding trace levels are SJ 1, SJ 2 and Exc. For more information about the trace points for the SJ domain, see the *CICS Trace Entries*.

The SJ domain also uses trace levels 29–32 to control JVM trace, that is, the trace produced by JVMs in the CICS region, rather than by the SJ domain itself. These levels correspond to trace levels 0, 1, and 2, plus a user-definable trace level, for JVMs. The JVM trace options are defined using a "free-form" 240–character field. "Controlling tracing for JVMs" in *Java Applications in CICS* tells you about the different ways to activate JVM trace and change the JVM trace options, and there is information about the meaning of the JVM trace options in the *IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.2 Diagnostics Guide*, SC34-6358, which is available to download from www.ibm.com/ developerworks/java/jdk/diagnosis/.

When you set trace levels 29–32 for the SJ component and activate JVM trace, the JVM trace appears as CICS trace point SJ 4D01. If the JVM trace facility fails, CICS issues the trace point SJ 4D00. Note that JVM trace can produce a large amount of output, so you should normally activate JVM trace for special transactions, rather than turning it on globally for all transactions.

For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 104. Storage manager domain (SM)

The storage manager domain (also sometimes known simply as “storage manager”) manages virtual storage requests.

Storage manager domain’s specific gates

Table 96 summarizes the storage manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 96. Storage manager domain’s specific gates

Gate	Trace	Function	XPI
SMAD	SM 0201	ADD_SUBPOOL	NO
	SM 0202	DELETE_SUBPOOL	NO
		START_SUBPOOL_BROWSE	NO
		GET_NEXT_SUBPOOL	NO
		END_SUBPOOL_BROWSE	NO
		INQUIRE_SUBPOOL	NO
SMAR	SM 0F01	ALLOCATE_TRANSACTION_STG	NO
	SM 0F02	RELEASE_TRANSACTION_STG	NO
SMCK	SM 0901	CHECK_STORAGE	NO
	SM 0902	RECOVER_STORAGE	NO
SMGF	SM 0301	GETMAIN	NO
	SM 0302	FREEMAIN	NO
		INQUIRE_ELEMENT_LENGTH	NO
SMMC	SM 0601	INITIALIZE	NO
	SM 0602		
	SM 0C01	GETMAIN	YES
	SM 0C02		
	SM 0D01	FREEMAIN	YES
	SM 0D02		
	SM 0E01	FREEMAIN_ALL_TERMINAL	NO
	SM 0E02		
	SM 0E01	INQUIRE_ELEMENT_LENGTH	YES
	SM 0E02		
		INQUIRE_TASK_STORAGE	YES
SMSR	SM 0401	INQUIRE_DSA_SIZE	NO
	SM 0402	SET_DSA_LIMIT	NO
		INQUIRE_DSA_LIMIT	NO
		SET_STORAGE_RECOVERY	NO
		SET_STORAGE_PROTECT	NO
		INQUIRE_STORAGE_PROTECT	NO
		INQUIRE_ACCESS_TOKEN	NO
		INQUIRE_ACCESS	YES
		SET_REENTRANT_PROGRAM	NO
		SET_TRANSACTION_ISOLATION	NO
		INQUIRE_REENTRANT_PROGRAM	NO
		INQUIRE_TRANSACTION_ISOLATION	NO
		SWITCH_SUBSPACE	YES
		INQUIRE_SHORT_ON_STORAGE	YES
		UPDATE_SUBSPACE_TCB_INFO	NO

SMAD gate, ADD_SUBPOOL function

The ADD_SUBPOOL function of the SMAD gate is used to create a new subpool with given attributes.

Input parameters

USAGE
indicates whether the subpool is for task or domain use. It can have either of these values:

`TASK|DOMAIN`

ELEMENT_TYPE
indicates whether the subpool elements are of fixed or variable length. It can have either of these values:

`FIXED|VARIABLE`

[FIXED_LENGTH]
is the element length for a fixed-length subpool.

ELEMENT_CHAIN
indicates whether a chain of the addresses and lengths of the elements is to be kept. It can have either of these values:

YES|NO

BOUNDARY
is the boundary on which all elements within the subpool must be aligned. The boundary must be a power of two in the range 8 through 4096.

LOCATION
specifies whether all elements within the subpool must be allocated below the maximum 24-bit address, or may be allocated anywhere. It can have either of these values:

BELOW|ANY

SUBPOOL_NAME
is the 8-character name by which the subpool is known.

INITIAL_FREE
is the size of the initial free storage area for the subpool.

[STORAGE_CHECK]
indicates whether storage zone checking is to be enabled for this subpool. It can have either of these values:

YES|NO

Output parameters

SUBPOOL_TOKEN
is the token identifying the newly created subpool.

[DSA_NAME]
is the name of the CICS dynamic storage area (DSA) in which the subpool resides. It can have any of these values:

CDSA|UDSA|SDSA|RDSA|ECDSA|EUDSA|ESDSA|ERDSA

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INSUFFICIENT_STORAGE
INVALID	INVALID_FIXED_LENGTH INVALID_BOUNDARY INVALID_SUBPOOL_NAME INVALID_INITIAL_FREE DUPLICATE_SUBPOOL_NAME

SMAD gate, DELETE_SUBPOOL function

The DELETE_SUBPOOL function of the SMAD gate is used to delete a subpool.

Input parameters

SUBPOOL_TOKEN
is the token identifying the subpool to be deleted.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_SUBPOOL_TOKEN NOT_SUBPOOL_OWNER SUBPOOL_NOT_EMPTY

SMAD gate, START_SUBPOOL_BROWSE function

The START_SUBPOOL_BROWSE function of the SMAD gate is used to start a browse of the storage manager domain subpools.

Input parameters

None

Output parameters

BROWSE_TOKEN is the token identifying the browse operation.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, with the following value:

RESPONSE	REASON value
EXCEPTION	INSUFFICIENT_STORAGE

SMAD gate, GET_NEXT_SUBPOOL function

The GET_NEXT_SUBPOOL function of the SMAD gate is used in a storage manager domain subpool browse to get the next subpool.

Input parameters

BROWSE_TOKEN is the token identifying the browse operation.

Output parameters

SUBPOOL_NAME is name of the subpool returned by the browse.

[DSA_NAME] is the name of the DSA in which the subpool resides. It can have one of the following values:

CDSA|UDSA|SDSA|RDSA|ECDSA|EUDSA|ESDSA|ERDSA

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, with the following value:

RESPONSE	REASON value
EXCEPTION	BROWSE_END

SMAD gate, END_SUBPOOL_BROWSE function

The END_SUBPOOL_BROWSE function of the SMAD gate is used to end a browse of the storage manager domain subpools.

Input parameters

BROWSE_TOKEN is the token identifying the browse operation.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] None.

SMAD gate, INQUIRE_SUBPOOL function

The INQUIRE_SUBPOOL function of the SMAD gate is used to inquire about a storage mangaer domain subpool.

Input parameters

SUBPOOL_NAME is the name of a storage manager domain subpool.

Output parameters

[DSA_NAME]
is the name of the DSA in which the subpool resides. It can have one of the following values:

CDSA|UDSA|SDSA|RDSA|ECDSA|EUDSA|ESDSA|ERDSA

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, with the following value:

RESPONSE	REASON value
EXCEPTION	SUBPOOL_NOT_FOUND

SMAR gate, ALLOCATE_TRANSACTION_STG function

The ALLOCATE_TRANSACTION_STG function of the SMAR gate is used at task initialization to add the four task lifetime storage subpools.

Input parameters

TASK_DATALOC
indicates the location of task data for the transaction, as specified by the TASKDATALOC attribute on the associated TRANSACTION resource definition. It can have either of these values:

BELOW|ANY

TASK_DATAKEY
indicates the storage key for the task-lifetime storage and program-related storage (for all programs that run under the transaction) for the transaction, as specified by the TASKDATAKEY attribute on the associated TRANSACTION resource definition. It can have either of these values:

CICS|USER

ISOLATE
indicates whether CICS is to isolate the transaction's user-key task-lifetime storage to provide application-to-application protection, as specified by the ISOLATE attribute on the associated TRANSACTION resource definition. It can have either of these values:

YES|NO

STORAGE_FREEZE
indicates whether or not task-lifetime storage freemains should be delayed until task termination. It can have either of these values:

YES|NO

STORAGE_CLEAR
indicates whether task lifetime storage should be cleared to zeros when it is freemained. It can have either of these values:

YES|NO

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND INSUFFICIENT_STORAGE

SMAR gate, RELEASE_TRANSACTION_STG function

The RELEASE_TRANSACTION_STG function of the SMAR gate is used at task termination to freemain all remaining task-lifetime storage and deletes the four task lifetime subpools.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND DEACTIVATE_FAILURE INSUFFICIENT_STORAGE STORAGE_VIOLATION

SMCK gate, CHECK_STORAGE function

The CHECK_STORAGE function of the SMCK gate is used to check the storage check zones of task lifetime storage and the storage accounting areas (SAAs) of terminal storage for consistency.

Input parameters

TASK_STORAGE
specifies whether the storage check zones of task lifetime storage are to be checked for the current task or all tasks, or is not to be checked. It can have any one of these values:

NO|CURRENT_TASK|ALL_TASKS

TP_STORAGE
specifies whether the SAAs of terminal storage are to be checked for the current terminal, or is not to be checked. It can have either of these values:

NO|CURRENT_TERMINAL

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP STORAGE_VIOLATION

SMCK gate, RECOVER_STORAGE function

The RECOVER_STORAGE function of the SMCK gate is used to recover storage.

Input parameters

TASK_STORAGE
specifies whether or not the task lifetime storage for the current task is to be recovered. It can have any one of these values:

NO|CURRENT_TASK

TP_STORAGE
specifies whether or not the SAAs of terminal storage for the current terminal are to be recovered. It can have either of these values:

NO|CURRENT_TERMINAL

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	STORAGE_NOT_RECOVERED

SMGF gate, GETMAIN function

The GETMAIN function of the SMGF gate is used to allocate an element of storage from a subpool.

Input parameters

Note: Either STORAGE_CLASS or SUBPOOL_TOKEN, but not both, must be specified.

[REMARK]
is an optional 8-character field that is used to identify the GETMAIN operation for problem determination. This field is highlighted when the GETMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being obtained.

[STORAGE_CLASS]
identifies the class of storage that is being allocated. It can have any one of these values:

CICS|CICS24|USER|USER24|TASK|TASK24

[SUBPOOL_TOKEN]
is a token identifying the subpool within which the element is to be allocated.

[GET_LENGTH]
is the length of the storage requested.

SUSPEND
If there is insufficient storage to satisfy the request, SUSPEND(YES) causes the caller to be suspended until the request can be satisfied, and SUSPEND(NO) causes REASON to be set to INSUFFICIENT_STORAGE. It can have either of these values:

YES|NO

[INITIAL_IMAGE]
is an optional byte value to which every byte in the new element is set.

Output parameters

ADDRESS
is the address of the new element.

[ELEMENT_LENGTH]
is the actual length of the new element (when it has been rounded up to a multiple of the boundary for the subpool).

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND ACTIVATE_FAILURE LOOP
EXCEPTION	INSUFFICIENT_STORAGE
INVALID	INVALID_SUBPOOL_TOKEN INVALID_GET_LENGTH INVALID_STORAGE_CLASS NO_GET_LENGTH NOT_SUBPOOL_OWNER INVALID_INITIAL_IMAGE

SMGF gate, FREEMAIN function

The FREEMAIN function of the SMGF gate is used to release an element of storage within a subpool.

Input parameters

Note: Either STORAGE_CLASS or SUBPOOL_TOKEN, but not both, must be specified.

[REMARK]
is an optional 8-character field that is used to identify the FREEMAIN operation for

problem determination. This field is highlighted when the FREEMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being released.

[STORAGE_CLASS]
identifies the class of storage that is being released. It can have any one of these values:

CICS|CICS24|USER|USER24|TASK|TASK24

[SUBPOOL_TOKEN]
is a token identifying the subpool within which the element is to be released.

ADDRESS
is the address of the element to be released.

[FREE_LENGTH]
is the length of the element to be released.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND DEACTIVATE_FAILURE LOOP
INVALID	INVALID_SUBPOOL_TOKEN INVALID_ADDRESS INVALID_FREE_LENGTH INVALID_STORAGE_CLASS NO_FREE_LENGTH NOT_SUBPOOL_OWNER SUBPOOL_EMPTY

SMGF gate, INQUIRE_ELEMENT_LENGTH function

The INQUIRE_ELEMENT_LENGTH function of the SMGF gate is used to return the length of an element of storage whose address is known.

Input parameters

Note: Either STORAGE_CLASS or SUBPOOL_TOKEN, but not both, must be specified.

[STORAGE_CLASS]
identifies the class of storage that is being inquired upon. It can have any one of these values:

CICS|CICS24|USER|USER24|TASK|TASK24

[SUBPOOL_TOKEN]
is a token identifying the subpool within which the element has been allocated.

ADDRESS
is the address of the element whose length is being inquired on.

Output parameters

ELEMENT_LENGTH
is the length of the element.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

RESPONSE	Possible REASON values
EXCEPTION	ADDRESS_NOT_FOUND
INVALID	INVALID_STORAGE_CLASS INVALID_SUBPOOL_TOKEN

SMMC gate, INQUIRE_ELEMENT_LENGTH function

The INQUIRE_ELEMENT_LENGTH function of the SMMC gate is used to obtain the start address and length of the storage element that contains the address that was specified on the input to the call. This function only searches the current task's task-lifetime storage for the required storage element.

Input parameters

ADDRESS
is the address to be searched for.

Output parameters

ELEMENT_LENGTH
is the length of the storage element that contains the input address.

[ELEMENT_ADDRESS]
is the start address of the element that contains the input address.

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	INVALID_ADDRESS

SMMC gate, INQUIRE_TASK_STORAGE function

The INQUIRE_TASK_STORAGE function of the SMMC gate is used to obtain details of all the task-lifetime storage associated with the current task (if the input parameter TRANSACTION_NUMBER is omitted from the call) or for the specified task.

Input parameters

[TRANSACTION_NUMBER]
indicates the transaction that you wish to obtain storage details about. If this parameter is omitted, the current task is assumed.

ELEMENT_BUFFER
is a buffer in which the storage manager lists the start addresses of all the specified task's task-lifetime storage.

LENGTH_BUFFER
is a buffer in which the storage manager lists the lengths of all the specified task's task-lifetime storage.

Output parameters

NUMBER_OF_ELEMENTS
is the number of elements in each buffer.

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

RESPONSE	Possible REASON values
EXCEPTION	INSUFFICIENT_STORAGE NO_TRANSACTION_ENVIRONMENT

SMMC gate, INITIALIZE function

The INITIALIZE function of the SMMC gate is used to perform macro-compatibility interface initialization.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMMC gate, GETMAIN function

The GETMAIN function of the SMMC gate is used to allocate an element of storage.

Input parameters

[REMARK]
is an optional 8-character field that is used to identify the GETMAIN operation for problem determination. This field is highlighted when the GETMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being obtained.

GET_LENGTH
is the length of storage requested. For storage classes that have 8-byte SAAs, the requested length excludes the lengths of the initial and duplicate SAAs. However, for storage classes that have only a 4-byte SAA, the requested length must include the length of the SAA.

SUSPEND
specifies whether the request is to be suspended if there is insufficient storage to satisfy the request. It can have either of these values:

`YES|NO`

[INITIAL_IMAGE]
specifies a byte value to which the user's part of the allocated storage element is to be set.

[TCTTE_ADDRESS]
is an optional field that must be specified for GETMAIN requests for the TERMINAL storage class.

STORAGE_CLASS
is the class of storage to be allocated. It can have any one of these values:

`TERMINAL24|CICS|SHARED_CICS|LINE|TERMINAL|`
`TASK|TASK24|CICS24_SAA|SHARED_CICS24_SAA|`
`CICS24|TRANSDATA|TEMPSTG|USER|USER24|`
`SHARED_CICS24|CONTROL|TACLE|SHARED_USER24|`
`SHARED_USER`

[CALLER]
can have any one of these values:

`EXEC|MACRO|SYSTEM`

Output parameters

ADDRESS
is the address of the allocated storage.

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND ACTIVATE_FAILURE LOOP
EXCEPTION	INSUFFICIENT_STORAGE INVALID_GET_LENGTH
INVALID	NO_TCTTE_ADDRESS INVALID_STORAGE_CLASS

SMMC gate, FREEMAIN function

The FREEMAIN function of the SMMC gate is used to release an element of storage.

Input parameters

[REMARK] is an optional 8-character field that is used to identify the FREEMAIN operation for problem determination. This field is highlighted when the FREEMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being released.

ADDRESS is the address of the storage to be freed.

[TCTTE_ADDRESS]
is an optional field that must be specified if the FREEMAIN is for storage of a LINE or TERMINAL class.

[STORAGE_CLASS]
is an optional field specifying the class of storage that is being freed. It can have any one of these values:

```
TERMINAL24|CICS|SHARED_CICS|LINE|TERMINAL|
TASK|TASK24|CICS24_SAA|SHARED_CICS24_SAA|
CICS24|TRANSDATA|TEMPSTG|USER|USER24|
SHARED_CICS24|CONTROL|TACLE|SHARED_USER24|
SHARED_USER
```

[CALLER] can have any one of these values:

```
EXEC|MACRO|SYSTEM
```

[EXEC_KEY] is the execution key of the program issuing the EXEC FREEMAIN request. It can have either of these values:

```
CICS|USER
```

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND DEACTIVATE_FAILURE LOOP
EXCEPTION	INVALID_EXEC_KEY
INVALID	INVALID_ADDRESS NO_TCTTE_ADDRESS

SMMC gate, FREEMAIN_ALL_TERMINAL function

The FREEMAIN_ALL_TERMINAL function of the SMMC gate is used to release all terminal storage.

Input parameters

TCTTE_ADDRESS
is the address of the TCTTE whose storage is to be freed.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMSR gate, INQUIRE_ISOLATION_TOKEN function

The INQUIRE_ISOLATION_TOKEN function of the SMSR gate is used to return an isolation token which can be used on SWITCH_SUBSPACE calls.

Input parameters

None.

Output parameters

ISOLATION_TOKEN
an isolation token which can be used on SWITCH_SUBSPACE calls.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMSR gate, INQUIRE_REENTRANT_PROGRAM function

The INQUIRE_REENTRANT_PROGRAM function of the SMSR gate is used to return whether the read-only DSAs, RDSA and ERDSA, have been allocated from read-only key-0 protected storage or CICS-key storage, as set by the RENTPGM system initialization parameter.

Input parameters

None.

Output parameters

RENTPGM indicates whether CICS has obtained the storage for the read-only DSAs from key-0 non-fetch protected storage (PROTECT) or from CICS-key storage (NOPROTECT). It can have either of these values:

PROTECT|NOPROTECT

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMSR gate, INQUIRE_SHORT_ON_STORAGE function

The INQUIRE_SHORT_ON_STORAGE function of the SMSR gate is used to return whether or not CICS is currently short-on-storage.

Input parameters

None.

Output parameters

SOS_BELOW_THE_LINE
: indicates whether or not CICS is short-on-storage below the 16MB line. It can have either of these values:

  YES|NO

SOS_ABOVE_THE_LINE
: indicates whether or not CICS is short-on-storage above the 16MB line. It can have either of these values:

  YES|NO

RESPONSE
: is the domain's response to the call. It can have any of these values:

  OK|DISASTER|INVALID|KERNERROR

[REASON]
: is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMSR gate, INQUIRE_DSA_SIZE function

The INQUIRE_DSA_SIZE function of the SMSR gate is used to return the size of the CICS DSAs.

Input parameters

DSA_NAME
: is the name of the DSA whose size is being inquired on. It can have any of these values:

  CDSA|UDSA|SDSA|RDSA|ECDSA|EUDSA|ESDSA|ERDSA

Output parameters

DSA_SIZE
: is the size of the DSA.

RESPONSE
: is the domain's response to the call. It can have any of these values:

  OK|DISASTER|INVALID|KERNERROR

[REASON]
: is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMSR gate, SET_STORAGE_RECOVERY function

The SET_STORAGE_RECOVERY function of the SMSR gate is used to set the storage recovery option.

Input parameters

RECOVERY
: is the value to which the storage recovery option is to be set. It can have either of these values:

  YES|NO

Output parameters

RESPONSE
: is the domain's response to the call. It can have any of these values:

  OK|DISASTER|INVALID|KERNERROR

[REASON]
: is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMSR gate, SET_TRANSACTION_ISOLATION function

The SET_TRANSACTION_ISOLATION function of the SMSR gate is used to set whether or not you want transaction isolation in your CICS region. This value is initially set by the TRANISO system initialization parameter.

Input parameters

TRANSACTION_ISOLATION
indicates whether or not transaction isolation is active in your CICS region. It can have either of these values:

ACTIVE|INACTIVE

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMSR gate, SWITCH_SUBSPACE function

The SWITCH_SUBSPACE function of the SMSR gate is used to change a task's subspace.

Input parameters

SPACE
indicates the type of subspace you wish this task to execute in. It can have any of these the values:

BASESPACE|SUBSPACE|RESET_SPACE

[ISOLATION_TOKEN]
an isolation token which can be returned from an INQUIRE_ISOLATION_TOKEN call.

[TRANSACTION_TOKEN]
a transaction manager token (which can be returned from an XMIQ INQUIRE_TRANSACTION_TOKEN call) that represents the task whose subspace you wish to change.

[WORK_REGISTER]
a work register.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMSR gate, INQUIRE_DSA_LIMIT function

The INQUIRE_DSA_LIMIT function of the SMSR gate is used to return the DSA storage limits above (EDSA) and below (DSA) the 16MB line. These limits are the maximum amounts of storage that CICS can use for *all* the DSAs above and below the 16MB line.

Input parameters

None.

Output parameters

[DSA_LIMIT] indicates the DSA storage limit.

[EDSA_LIMIT] indicates the EDSA storage limit.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMSR gate, SET_DSA_LIMIT function

The SET_DSA_LIMIT function of the SMSR gate is used to set the DSA storage limits above (EDSA) and below (DSA) the 16MB line. These limits are the maximum amounts of storage that CICS can use for *all* the DSAs above and below the 16MB line.

Input parameters

[DSA_LIMIT] indicates the DSA storage limit required.

[EDSA_LIMIT] indicates the EDSA storage limit required.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	INSUFFICIENT_STORAGE INVALID_DSA_LIMIT

SMSR gate, SET_STORAGE_PROTECT function

The SET_STORAGE_PROTECT function of the SMSR gate is used to set the storage protection option.

Input parameters

STORAGE_PROTECT

can have either of these values:

YES|NO

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	NO_HARDWARE_SUPPORT

SMSR gate, INQUIRE_STORAGE_PROTECT function

The INQUIRE_STORAGE_PROTECT function of the SMSR gate is used to return the current value of the storage protection option.

Input parameters

None.

Output parameters

STORAGE_PROTECT
is the current storage protection mode. It can have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMSR gate, INQUIRE_ACCESS function

The INQUIRE_ACCESS function of the SMSR gate is used to return the access key of an element of storage.

Input parameters

[ACCESS_TOKEN]
is the access token for the element of storage (returned by the INQUIRE_ACCESS_TOKEN function).

ELEMENT_ADDRESS
is the start address of the storage element.

ELEMENT_LENGTH
is the length of the storage element.

Output parameters

ACCESS
is the type of access for the storage element. It can have any of these values:

CICS|USER|READ_ONLY

[DSA_NAME]
is the name of the DSA in which the storage element resides.

[DSA_EXTENT_START]
indicates the start address of the DSA extent that contains the input address.

[DSA_EXTENT_END]
indicates the end address of the DSA extent that contains the input address.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	INVALID_ELEMENT

SMSR gate, SET_REENTRANT_PROGRAM function

The SET_REENTRANT_PROGRAM function of the SMSR gate is used to set the reentrant program option for the RDSA and the ERDSA.

Input parameters

REENTRANT_PROGRAM
is the reentrant program option for the RDSA and the ERDSA. It can have either of these values:

PROTECT|NOPROTECT

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMSR gate, INQUIRE_ACCESS_TOKEN function

The INQUIRE_ACCESS_TOKEN function of the SMSR gate is used to return the access token for a storage element.

Input parameters

None.

Output parameters

ACCESS_TOKEN
is the access token for the storage element.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

SMSR gate, UPDATE_SUBSPACE_TCB_INFO function

The UPDATE_SUBSPACE_TCB_INFO function informs SM of the deletion of open TCBs which are associated with subspaces.

Input parameters

SUBSPACE_TOKEN
indicates the subspace which is associated with the deleted TCBs.

OPEN_TCBS_DELETED
is a 32-bit string indicating the mode(s) of deleted TCB(s).

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR

Storage manager domain’s generic gates

Table 97 summarizes the storage manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 97. Storage manager domain’s generic gates

Gate	Trace	Function	Format
DMDM	SM 0101 SM 0102	PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
SMVN	SM 1401 SM 1402	DSAT_TASK_REPLY DSAT_PURGE_INHIBIT_QUERY	DSAT
STST	SM 0A01 SM 0A02	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—“Domain manager domain’s generic formats” on page 669

Format DSAT—“DSAT format, TASK_REPLY function” on page 717

Format STST—“Statistics domain’s generic format” on page 1198

In preinitialization processing, the storage manager domain sets the initial storage options:

- The amount of storage to be allocated to the dynamic storage area
- The amount of storage to be allocated to the extended dynamic storage area
- The storage recovery option
- The state of the storage protect, transaction isolation and the reentrant program option.

For a cold start, the information comes from the system initialization parameters; for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

Storage manager domain also issues console messages during preinitialization to report the amount of storage allocated above and below the line for DSA use.

In initialization, quiesce, and termination processing, the storage manager domain performs only internal routines.

Storage manager domain’s generic formats

Table 98 shows the generic formats owned by the storage manager domain, and shows the functions performed on the calls.

Table 98. Generic formats owned by the storage manager domain

Format	Calling module	Function
SMNT	DFHSMSY DFHSJSM	STORAGE_NOTIFY MVS_STORAGE_NOTIFY

In the descriptions of the formats that follow, the “input” parameters are input not to the storage manager domain, but to the domain being called by the storage manager. Similarly, the “output” parameters are output by the domain that was called by the storage manager domain, in response to the call.

Format SMNT, STORAGE_NOTIFY function

The STORAGE_NOTIFY function of SMNT format is used to notify free storage above and below the 16MB line.

Input parameters

DSAS_CONSTRAINED YES|NO
: indicates whether any DSA is currently constrained due to lack of free storage.

FREE_BYTES_CDSA
: is the largest free area available (in bytes) in the CICS DSA below the 16MB line (not including the cushion).

FREE_BYTES_UDSA
: is the largest free area available (in bytes) in the user-key DSA below the 16MB line (not including the cushion).

FREE_BYTES_SDSA
: is the largest free area available (in bytes) in the shared user-key DSA below the 16MB line (not including the cushion).

FREE_BYTES_RDSA
: is the largest free area available (in bytes) in the read-only DSA below the 16MB line (not including the cushion).

FREE_BYTES_ECDSA
: is the largest free area available (in bytes) in the CICS DSA above the 16MB line (not including the cushion).

FREE_BYTES_EUDSA
: is the largest free area available (in bytes) in the user-key DSA above the 16MB line (not including the cushion).

FREE_BYTES_ESDSA
: is the largest free area available (in bytes) in the shared user-key DSA above the 16MB line (not including the cushion).

FREE_BYTES_ERDSA
: is the largest free area available (in bytes) in the read-only DSA above the 16MB line (not including the cushion).

Output parameters

RESPONSE
: is the domain’s response to the call. It can have any of these values:

  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
: is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOOP ABEND
INVALID	INVALID_FUNCTION

Format SMNT, MVS_STORAGE_NOTIFY function

The MVS_STORAGE_NOTIFY function of SMNT format is used to notify a domain when MVS storage usage becomes excessive, so that the target domain can take action to release MVS storage or to limit its future MVS storage requirements. It is also used to notify the domain when MVS storage is no longer constrained, so the domain can return to normal operation. There are different notifications for a breach of the threshold value for MVS storage, and for a breach of the reserved MVS storage cushion, the latter being a more serious condition.

Input parameters

THRESHOLD
indicates the relationship between MVS storage requirements and the threshold value for MVS storage. It can have the values:

NEWLY_BREACHED|NEWLY_RESTORED|UNCHANGED

NEWLY_BREACHED indicates that MVS storage requirements have increased above the threshold value since the last time the SM domain issued a MVS_STORAGE_NOTIFY. NEWLY_RESTORED indicates that MVS storage requirements have decreased below the threshold value since the last time the SM domain issued a MVS_STORAGE_NOTIFY. UNCHANGED indicates that since the last time the SM domain issued a MVS_STORAGE_NOTIFY, no change has occurred in the MVS storage requirements relative to the threshold value. That is, if the MVS storage requirements were previously above the threshold, they are still above the threshold, and if they were previously below the threshold, they are still below the threshold.

CUSHION
indicates the status of the reserved MVS storage cushion. It can have the values:

NEWLY_BREACHED|NEWLY_RESTORED|UNCHANGED

NEWLY_BREACHED indicates that the cushion has been partially freed to satisfy requirements for MVS storage since the last time the SM domain issued a MVS_STORAGE_NOTIFY. NEWLY_RESTORED indicates that CICS has managed to reallocate the reserved storage cushion since the last time the SM domain issued a MVS_STORAGE_NOTIFY. UNCHANGED indicates that since the last time the SM domain issued a MVS_STORAGE_NOTIFY, no change has occurred in the state of the cushion: it is still partially freed, or still intact.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOOP ABEND LOCK_FAILED RESUME_FAILURE
INVALID	INVALID_FUNCTION

Modules

Module	Function
DFHSMAD	Handles the following requests: ADD_SUBPOOL DELETE_SUBPOOL START_SUBPOOL_BROWSE GET_NEXT_SUBPOOL END_SUBPOOL_BROWSE INQUIRE_SUBPOOL
DFHSMAR	Handles the following requests: ALLOCATE_TRANSACTION_STG RELEASE_TRANSACTION_STG
DFHSMCK	Handles the following requests: CHECK_STORAGE RECOVER_STORAGE

Module	Function
DFHSMDM	Handles the following requests: PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHSMDUF	SM domain offline dump formatting routine
DFHSMGF	Handles the following requests: GETMAIN FREEMAIN INQUIRE_ELEMENT_LENGTH
DFHSMMCI	SM domain macro-compatibility interface INITIALISE function
DFHSMMC2	SM domain macro-compatibility interface which handles the following requests: FREEMAIN_ALL_TERMINAL INQUIRE_ELEMENT_LENGTH INQUIRE_TASK_STORAGE
DFHSMMF	SM domain macro-compatibility interface FREEMAIN function
DFHSMMG	SM domain macro-compatibility interface GETMAIN function
DFHSMSR	Handles the following requests: INQUIRE_ACCESS INQUIRE_ACCESS_TOKEN INQUIRE_DSA_LIMIT INQUIRE_DSA_SIZE INQUIRE_REENTRANT_PROGRAM INQUIRE_SHORT_ON_STORAGE INQUIRE_STORAGE_PROTECT INQUIRE_TRANSACTION_ISOLATION SET_DSA_LIMIT SET_REENTRANT_PROGRAM SET_STORAGE_RECOVERY SET_STORAGE_PROTECT SWITCH_SUBSPACE
DFHSMST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHSMSVC	Gets DSAs
DFHSMSY	SM domain system task—issues STORAGE_NOTIFY requests
DFHSMTRI	Interprets SM domain trace entries
DFHSMVN	SM domain system task — issues MVS_STORAGE_NOTIFY requests
DFHSMVP	Detects and manages MVS storage constraints

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the storage manager domain are of the form SM xxxx; the corresponding trace levels are SM 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 105. Sign-on component

The sign-on routine is a component of terminal control that associates users with terminals, connections, and sessions. Because it is a part of terminal control, it is logically part of the application (AP) domain.

Sign-on component's subroutines

The sign-on component is entered as a single kernel-managed subroutine, DFHSNUS, which handles some function itself and also acts as a router to further kernel-managed subroutines. Table 99 summarizes the sign-on component's subroutines. It shows the level-1 trace point IDs of the modules providing the functions for the subroutines, the functions provided by the subroutines, and whether or not the functions are available through the exit programming interface (XPI).

Table 99. Sign-on component's subroutines

Subroutine	Trace	Function	XPI
DFHSNAS	AP 2050 AP 2051 AP 2052 AP 2053 AP 2054 AP 2055 AP 2056	SIGNON_ATI_SESSION SIGNOFF_ATI_SESSION	NO NO
DFHSNPU	AP 2070 AP 2071 AP 2072 AP 2073 AP 2074 AP 2075 AP 2076 AP 2077 AP 2078 AP 2079	SIGNON_PRESET_USERID SIGNOFF_PRESET_USERID	NO NO
DFHSNSG	AP 20C0 AP 20C1 AP 20C2 AP 20C3 AP 20C4 AP 20C5 AP 20C6	SIGNOFF_SURROGATE	NO
DFHSNSU	AP 2060 AP 2061 AP 2062 AP 2063 AP 2064 AP 2065 AP 2066 AP 2067 AP 2068 AP 2069 AP 206A AP 206B AP 206C AP 206D	SIGNON_SESSION_USERID SIGNOFF_SESSION_USERID	NO NO

Table 99. Sign-on component's subroutines (continued)

Subroutine	Trace	Function	XPI
DFHSNTU	AP 2080 AP 2081 AP 2082 AP 2083 AP 2084 AP 2085 AP 2086 AP 2087 AP 2088 AP 2089 AP 208A AP 208B AP 208C AP 208D AP 208E AP 208F AP 2090 AP 2091 AP 2092 AP 2093 AP 2094 AP 2095 AP 2096 AP 2097	SIGNON_TERMINAL_USER SIGNOFF_TERMINAL_USER	NO NO
DFHSNUS	AP 2040 AP 2041 AP 2042 AP 2043 AP 2044 AP 2045 AP 2046 AP 2047 AP 2048 AP 2049	SIGNON_ATTACH_HEADER SIGNOFF_ATTACH_HEADER	NO NO

DFHSNAS subroutine, SIGNON_ATI_SESSION function

The SIGNON_ATI_SESSION function of the DFHSNAS subroutine signs on the appropriate userid to a session when that session is being used by a trigger transaction specified in a DCT with DESTFAC=SYSTEM.

Input parameters

SESSION_TCTTE_PTR
is the address of the TCTTE for the session to be signed on.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP UNEXPECTED_REASON CORRUPT_USER_TOKEN USER_DOMAIN_FAILURE USER_TOKEN_MISMATCH
EXCEPTION	INVALID_TERMINAL_TYPE TERMINAL_ALREADY_SIGNED_ON SURROGATE_TERMINAL SECURITY_INACTIVE
INVALID	INVALID_FORMAT INVALID_FUNCTION

DFHSNAS subroutine, SIGNOFF_ATI_SESSION function

The SIGNOFF_ATI_SESSION function of the DFHSNAS subroutine is used to reverse the effect of a SIGNON_ATI_SESSION.

Input parameters

SESSION_TCTTE_PTR
is the address of the session TCTTE to be signed off.

Output parameters

RESPONSE
is the subroutine's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP CORRUPT_USER_TOKEN USER_DOMAIN_FAILURE INVALID_USER_TOKEN
EXCEPTION	INVALID_TERMINAL_TYPE TERMINAL_NOT_SIGNED_ON SURROGATE_TERMINAL SECURITY_INACTIVE
INVALID	INVALID_FORMAT INVALID_FUNCTION

DFHSNPU subroutine, SIGNON_PRESET_USERID function

The SIGNON_PRESET_USERID function of the DFHSNPU subroutine is used to sign on the userid specified in a terminal definition when that terminal is installed.

Input parameters

USERID
is the userid to be assigned to the terminal.

USERID_LENGTH
is the length of the userid.

TCTTE_PTR
is the address of the TCTTE for the terminal to be given preset security.

[NATLANG_SUFFIX]
is an optional one-character national language code to be assigned to the terminal, which will override any national language associated with the userid.

[MESSAGE] is an optional parameter that specifies whether a message is to be issued when the sign on completes successfully. It can have either of these values:

YES|NO

Output parameters

[SAF_RESPONSE] is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE] is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the subroutine's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP EXCEPTION_UNKNOWN CORRUPT_USER_TOKEN MESSAGE_DOMAIN_FAILURE USER_DOMAIN_FAILURE GETMAIN_FAILED
EXCEPTION	INVALID_USERID INVALID_NATIONAL_LANGUAGE TERMINAL_ALREADY_SIGNED_ON UNKNOWN_ESM_RESPONSE SECURITY_INACTIVE ESM_INACTIVE TERMINAL_NOTAUTH APPLICATION_NOTAUTH USERID_REVOKED TERMINAL_NOT_PRESET GROUP_ACCESS_REVOKED UNAVAILABLE_NATLANG SECLABEL_CHECK_FAILED ESM_TRANQUIL
INVALID	INVALID_FORMAT INVALID_FUNCTION

DFHSNPU subroutine, SIGNOFF_PRESET_USERID function

The SIGNOFF_PRESET_USERID function of the DFHSNPU subroutine is used to sign off a preallocated userid from a terminal before it is deleted.

Input parameters

TCTTE_PTR is the address of the TCTTE for the terminal from which preset security is to be removed.

[MESSAGE] is an optional parameter that specifies whether a message is to be issued when the sign off completes successfully. It can have either of these values:

YES|NO

Output parameters

[SAF_RESPONSE] is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE] is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the subroutine's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP CORRUPT_USER_TOKEN FREEMAIN_FAILED
EXCEPTION	TERMINAL_NOT_SIGNED_ON TERMINAL_NOT_PRESET SECURITY_INACTIVE ESM_INACTIVE ESM_TRANQUIL UNKNOWN_ESM_RESPONSE
INVALID	INVALID_FORMAT INVALID_FUNCTION

DFHSNSG subroutine, SIGNOFF_SURROGATE function

The SIGNOFF_SURROGATE function of the DFHSNSG subroutine is used to sign off a userid from a surrogate terminal that is about to be deleted by the remote terminal builder. (The equivalent sign-on routine is always performed as an inline function, so no SIGNON call to DFHSNSG is ever traced.)

Input parameters

TCTTE_PTR is the address of the TCTTE for the surrogate terminal being signed off.

SESSION_TCTTE_PTR is the address of the TCTTE for the associated relay session.

Output parameters

RESPONSE is the subroutine's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

DFHSNSU subroutine, SIGNON_SESSION_USERID function

The SIGNON_SESSION_USERID function of the DFHSNSU subroutine is used to sign on the USERID (from the SESSIONS definition) or the SECURITYNAME (from the CONNECTION definition) for IRC, LU6.1, and LU6.2 sessions.

Input parameters

[USERID] is the userid to be signed on.

[USERID_LENGTH] is the length of the userid to be signed on.

SESSION_TCTTE_PTR is the address of the TCTTE for the session being signed on.

Output parameters

RESPONSE is the subroutine's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP USER_TOKEN_MISMATCH MESSAGE_DOMAIN_FAILURE SURROGATE_TERMINAL USER_DOMAIN_FAILURE XS_DOMAIN_FAILURE
EXCEPTION	INVALID_USERID INVALID_TERMINAL_TYPE TERMINAL_ALREADY_SIGNED_ON UNKNOWN_ESM_RESPONSE SECURITY_INACTIVE ESM_INACTIVE APPLICATION_NOTAUTH USERID_REVOKED GROUP_ACCESS_REVOKED SECLABEL_CHECK_FAILED ESM_TRANQUIL
INVALID	INVALID_FORMAT INVALID_FUNCTION

DFHSNSU subroutine, SIGNOFF_SESSION_USERID function

The SIGNOFF_SESSION_USERID function of the DFHSNSU subroutine is used to reverse the effect of the SIGNON_SESSION_USERID function.

Input parameters

None

Output parameters

[SAF_RESPONSE] is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE] is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the subroutine's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP TERMINAL_NOT_SIGNED_ON CORRUPT_USER_TOKEN INVALID_TERMINAL_TYPE SURROGATE_TERMINAL SECOND_DELETE_FAILED MESSAGE_DOMAIN_FAILURE USER_DOMAIN_FAILURE
EXCEPTION	SECURITY_INACTIVE ESM_INACTIVE ESM_TRANQUIL UNKNOWN_ESM_RESPONSE
INVALID	INVALID_FORMAT INVALID_FUNCTION

DFHSNTU subroutine, SIGNON_TERMINAL_USER function

The SIGNON_TERMINAL_USER function of the DFHSNTU subroutine is used to implement the EXEC CICS SIGNON command and signs on a specific user at the principal facility terminal.

Input parameters

USERID
is the userid being signed on to the principal facility terminal.

USERID_LENGTH
is the length of the userid.

[PASSWORD]
is the optional password associated with the userid. The external security manager determines whether the password is required or not.

[PASSWORD_LENGTH]
is the length of the password.

[NEW_PASSWORD]
is the optional new password that is to replace the existing password

[NEW_PASSWORD_LENGTH]
is the length of the new password.

[OIDCARD]
is the text obtained from an operator identification card. The external security manager determines whether operator identification card data, or a password, or both, or neither, are required.

[GROUPID]
is the optional group name to be associated with the userid for this sign on.

[GROUPID_LENGTH]
is the length of the group name.

[NATIONAL_LANGUAGE]
is the optional three-letter national language code to be associated with the terminal for the duration of this sign on. The code should be one of those specified in Table 120 on page 1318.

[SCOPE_CHECK]
is an optional parameter that specifies whether this sign on is to be subject to the constraints imposed by the SNSCOPE system initialization parameter. It can have either of these values:

YES|NO

Output parameters

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the subroutine's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP ADD_USER_FAILURE GETMAIN_FAILED EXCEPTION_UNKNOWN INQUIRE_DEFAULT_ERROR MESSAGE_DOMAIN_FAILURE USER_DOMAIN_FAILURE XMIQ_FAILURE CORRUPT_USER_TOKEN SNXR_FAILURE SUSX_FAILURE

RESPONSE	Possible REASON values
EXCEPTION	INVALID_USERID INVALID_PASSWORD INVALID_NEW_PASSWORD INVALID_OIDCARD INVALID_GROUPID USERID_NOT_IN_GROUP INVALID_TERMINAL_TYPE INVALID_NATIONAL_LANGUAGE UNAVAILABLE_NATLANG TERMINAL_ALREADY_SIGNED_ON USERID_ALREADY_SIGNED_ON SURROGATE_TERMINAL PRESET_SECURITY_TERMINAL NO_TERMINAL_WITH_TASK USERID_REQUIRED PASSWORD_REQUIRED NEW_PASSWORD_REQUIRED OIDCARD_REQUIRED UNKNOWN_ESM_RESPONSE SECURITY_INACTIVE ESM_INACTIVE TERMINAL_NOTAUTH APPLICATION_NOTAUTH USERID_REVOKED GROUP_ACCESS_REVOKED SECLABEL_CHECK_FAILED ESM_TRANQUIL
INVALID	INVALID_FORMAT INVALID_FUNCTION

DFHSNTU subroutine, SIGNOFF_TERMINAL_USER function

The SIGNOFF_TERMINAL_USER function of the DFHSNTU subroutine is used to implement the EXEC CICS SIGNOFF command and reverses the effect of a SIGNON_TERMINAL_USER function. It effectively associates the terminal with the default userid specified in the DFLTUSER system initialization parameter.

Input parameters

[TCTTE_PTR]
is the optional TCTTE address of a terminal that is to be signed off.

Output parameters

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the subroutine's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND FREEMAIN_FAILED LOOP XMIQ_FAILURE ADD_TXN_USER_ERROR INVALID_USER_TOKEN

RESPONSE	Possible REASON values
EXCEPTION	INVALID_TERMINAL_TYPE TERMINAL_NOT_SIGNED_ON PRESET_SECURITY_TERMINAL SURROGATE_TERMINAL NO_TERMINAL_WITH_TASK SECURITY_INACTIVE ESM_INACTIVE ESM_TRANQUIL UNKNOWN_ESM_RESPONSE
INVALID	INVALID_FORMAT INVALID_FUNCTION

DFHSNUS subroutine, SIGNON_ATTACH_HEADER function

The SIGNON_ATTACH_HEADER function of the DFHSNUS subroutine causes a sign on for the userid received in an LU6.2 function management header type 5, also known as an *attach header* or an FMH5.

Input parameters

TCTTE_PTR
is the address of the TCTTE for which the FMH5 sign on is being performed.

[USERID]
is the userid obtained from the FMH5, if any.

[USERID_LENGTH]
is the length of the userid

[PASSWORD]
is the password obtained fron the FMH5, if any.

[PASSWORD_LENGTH]
is the length of the password.

[GROUPID]
is the group name obtained from the profile name in the FMH5, if any.

[GROUPID_LENGTH]
is the length of the group name.

[ENTRY_PORT_NAME]
is the optional name of the entry port (terminal) at which the userid was signed on in the terminal-owning region.

[ENTRY_PORT_TYPE]
is the optional terminal type associated with the port of entry. It can have either of these values:

TERMINAL|CONSOLE

[APPLID]
is the optional applid at which the userid was signed on in the terminal-owning region.

ATTACHSEC_TYPE
specifies whether the ATTACHSEC associated with the connection is LOCAL or not. It can have either of these values:

LOCAL|NON_LOCAL

ALREADY_VERIFIED
specifies whether the already-verified indicator (AV) is present in the FMH5. It can have either of these values:

YES|NO

PERSISTENT_SIGNON
specifies whether the persistent-sign on indicator (PV2) is present in the FMH5. It can have either of these values:

YES|NO

PERSISTENT_VERIFY
specifies whether the persistent-verification indicator (PV1) is present in the FMH5. It can have either of these values:

YES|NO

Output parameters

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the subroutine's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP USER_DOMAIN_FAILURE MESSAGE_DOMAIN_FAILURE CORRUPT_USER_TOKEN ZCUT_FAILURE
EXCEPTION	TERMINAL_ALREADY_SIGNED_ON INVALID_USERID INVALID_PASSWORD INVALID_GROUPID USERID_NOT_IN_GROUP PRESET_SECURITY_TERMINAL USERID_REQUIRED PROTOCOL_VIOLATION PASSWORD_REQUIRED UNKNOWN_ESM_RESPONSE SECURITY_INACTIVE ESM_INACTIVE TERMINAL_NOTAUTH LUIT_ENTRY_NOT_FOUND APPLICATION_NOTAUTH USERID_REVOKED GROUP_ACCESS_REVOKED SECLABEL_CHECK_FAILED SIGNON_SURROGATE_ERROR ESM_TRANQUIL
INVALID	INVALID_FORMAT INVALID_FUNCTION

DFHSNUS subroutine, SIGNOFF_ATTACH_HEADER function

The SIGNOFF_ATTACH_HEADER function of the DFHSNUS subroutine is used to reverse the effect of a SIGNON_ATTACH_HEADER function when the transaction initiated by the FMH5 attach header terminates.

Input parameters

TCTTE_PTR is the address of the TCTTE for which the FMH5 sign off is being performed.

Output parameters

[SAF_RESPONSE] is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE] is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the subroutine’s response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP ZCUT_FAILURE CORRUPT_USER_TOKEN
EXCEPTION	INVALID_TERMINAL_TYPE PRESET_SECURITY_TERMINAL SURROGATE_TERMINAL SECURITY_INACTIVE ESM_INACTIVE ESM_TRANQUIL UNKNOWN_ESM_RESPONSE
INVALID	INVALID_FORMAT INVALID_FUNCTION

Modules

Module	Function
DFHSNAS	Handles the following requests: SIGNON_ATI_SESSION SIGNOFF_ATI_SESSION
DFHSNPU	Handles the following requests: SIGNON_PRESET_USERID SIGNOFF_PRESET_USERID
DFHSNDUF	SN domain offline dump formatting routine
DFHSNSG	Handles the following requests: SIGNOFF_SURROGATE
DFHSNSU	Handles the following requests: SIGNON_SESSION_USERID SIGNOFF_SESSION_USERID
DFHSNTU	Handles the following requests: SIGNON_TERMINAL_USER SIGNOFF_TERMINAL_USER
DFHSNUS	Acts as a router to the other signon modules, and handles the following requests directly: SIGNON_ATTACH_HEADER SIGNOFF_ATTACH_HEADER
DFHSNTRI	Interprets SN domain trace entries

Exits

There are two global user exit points in DFHSNUS: XSNON and XSNOFF.

For further information, see the *CICS Customization Guide*.

Trace

The point IDs for the sign on component are of the form AP xxxx; the corresponding trace levels are AP 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 106. Socket domain (SO)

The socket domain provides TCP/IP services to the CICS Web Support and CICS IIOP Support components. It includes a TCP/IP listener system task, the TCPIPSERVICE RDO resource to manage the listener and domain gates to operate on a TCP/IP connection.

Socket domain's specific gates

Table 100 summarizes the socket domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 100. Socket domain's specific gates

Gate	Trace	Function	XPI
SOCK	SO 0201 SO 0202	SEND SEND_SSL_DATA RECEIVE RECEIVE_SSL_DATA CLOSE LISTEN	NO NO NO NO NO NO
SORD	SO 0301 SO 0302	REGISTER DEREGISTER IMMCLOSE	NO NO NO
SOIS	SO 0401 SO 0402	SET_PARAMETERS INITIALIZE_ENVIRONMENT INQUIRE SET INQUIRE_STATISTICS VERIFY EXPORT_CERTIFICATE_DATA IMPORT_CERTIFICATE_DATA DELETE_CERTIFICATE_DATA	NO NO NO NO NO NO NO NO NO
SOAD	SO 0601 SO 0602	ADD_REPLACE_TCPIPSERVICE DELETE_TCPIPSERVICE	NO NO
SOTB	SO 0701 SO 0702	INQUIRE_TCPIPSERVICE START_BROWSE GET_NEXT END_BROWSE SET_TCPIPSERVICE	NO NO NO NO NO
SOSE	SO 0801 SO 0802	INITIALIZE_SSL SECURE_SOC_INIT SECURE_SOC_READ SECURE_SOC_WRITE SECURE_SOC_CLOSE SECURE_SOC_RESET TERMINATE_SSL EXPORT_CERTIFICATE_DATA IMPORT_CERTIFICATE_DATA DELETE_CERTIFICATE_DATA	NO NO NO NO NO NO NO NO NO NO

SOCK gate, SEND function

The SEND function sends a buffer of data to a connected TCP/IP client.

Input parameters

SEND_BUFFER
is the buffer of data to be sent.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_SESSION_TOKEN INSUFFICIENT_STORAGE IO_ERROR,CONNECTION_CLOSED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE SOCKET_IN_USE

SOCK gate, SEND_SSL_DATA function

The SEND_SSL_DATA function is called to send data to a connected TCP/IP client if the connection is secured using SSL.

Input parameters

STE_PTR is a pointer to the STE control block of the session.

SEND_BUFFER is the buffer of data to be sent.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_SESSION_TOKEN INSUFFICIENT_STORAGE IO_ERROR,CONNECTION_CLOSED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE SOCKET_IN_USE

SOCK gate, RECEIVE function

The RECEIVE function receives a buffer of data from a TCP/IP connected client.

Input parameters

RECEIVE_BUFFER
is the buffer to receive the data into.

[TIMEOUT] is an optional parameter. It can take two values:

DEFAULT|SOCKETCLOSE

If not specified or a value of SOCKETCLOSE is specified then the timeout is taken from the tcpipservice definition. If DEFAULT is specified then the timeout is 30 seconds.

Output parameters

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_SESSION_TOKEN INSUFFICIENT_STORAGE IO_ERROR,CONNECTION_CLOSED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE SOCKET_IN_USE

SOCK gate, RECEIVE_SSL_DATA function

The RECEIVE_SSL_DATA function is called to receive data from a connected TCP/IP client if the connection is secured using SSL.

Input parameters

STE_PTR is a pointer to the STE control block of the session.

RECEIVE_BUFFER
is the buffer to receive data into.

Output parameters

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_SESSION_TOKEN INSUFFICIENT_STORAGE IO_ERROR,CONNECTION_CLOSED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE SOCKET_IN_USE

SOCK gate, CLOSE function

The CLOSE function is called to close the socket connection to the TCP/IP client.

Output parameters

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_SESSION_TOKEN INSUFFICIENT_STORAGE IO_ERROR,CONNECTION_CLOSED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE SOCKET_IN_USE

SOCK gate, LISTEN function

The LISTEN function is the main routine for the SO domain listener task CSOL. When the listener task starts it branches into the LISTEN function of the SOCK gate. This allows the listener code to be written at the domain level rather than the task level.

Output parameters

RESPONSE
is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SORD gate, REGISTER function

The REGISTER function is called to open a tcpipservice. It registers all the parameters of the service with the listener task.

Input parameters

PORT_NUMBER
is the TCP/IP port number to listen for new connection on.

SERVICE_NAME
is the name of the tcpipservice.

TRANID
is the transaction ID that is to be attached when a new connection is made to the listening port.

SSL
specifies whether or not connections to this service are to be secured using the Secure Sockets Layer protcols.

BACKLOG
is the value of the backlog parameter passed to the TCP/IP listen function for this service. It specifies how many connection requests TCP/IP will queue for this service.

URM
is the name of a user-replacable program that the handler transaction for this service will invoke during request processing.

TSQPREFIX
is the prefix for TS queues that are created by the programs handling requests for this service.

IPADDRESS
is the specific IP address that the listener will bind to for this service.

[CERTIFICATE_LABEL]
is the name of a certificate within the keyfile that this service will use to authenticate itself to clients with, if the SSL protocol is used.

RECV_TIMEOUT
specifies whether or not receives should timeout, and if so, after how long.

Output parameters

LISTEN_TOKEN is a token representing the opened tcpipservice. This is subsequently used to close the service.

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_LISTEN_TOKEN INSUFFICIENT_STORAGE NOT_PERMITTED_TO_BIND TCPIP_SERVICE_ERROR TCPIP_CLOSED TCPIP_INACTIVE UNKNOWN_ADDRESS PORT_IN_USE
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SORD gate, DEREGISTER function

The DEREGISTER function is called to close a tcpipservice. The listener task closes the listening socket and no more connections to the port are permitted. Any tasks handling existing connections are allowed to end normally.

Input parameters

LISTEN_TOKEN is a token representing the opened tcpipservice.

Output parameters

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_LISTEN_TOKEN INSUFFICIENT_STORAGE NOT_PERMITTED_TO_BIND TCPIP_SERVICE_ERROR TCPIP_CLOSED TCPIP_INACTIVE UNKNOWN_ADDRESS, PORT_IN_USE
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SORD gate, IMMCLOSE function

The IMMCLOSE function is called to immediatly close a tcpipservice. The listener task closes the listening socket and no more connections to the port are permitted. All existing connections are closes and any tasks handling them are abended.

Input parameters

LISTEN_TOKEN is a token representing the opened tcpipservice.

Output parameters

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_LISTEN_TOKEN INSUFFICIENT_STORAGE NOT_PERMITTED_TO_BIND TCPIP_SERVICE_ERROR TCPIP_CLOSED TCPIP_INACTIVE UNKNOWN_ADDRESS, PORT_IN_USE
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOIS gate, SET_PARAMETERS function

The SET_PARAMETERS function is called during CICS initialisation when the SIT is processed. It sets the startup parameters for the SO domain.

Input parameters

TCPIP is a YES or NO value indicating if the SO domain is to initalise in this CICS region.

SSLDELAY is a the SSL timeout value.

SSLTCBS specifies the number of S8 TCBs to be attached for SSL use.

ENCRYPTION specifies the type of encryption that will be used by the system. The value can be NORMAL,STRONG or WEAK.

KEYFILE specifies the name of the HFS keyring file that contains the keypairs and certificate data.

QUALIFIER is actually the password that was used to secure the keyring file upon creation.

Output parameters

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOIS gate, INITIALIZE_ENVIRONMENT function

The INITIALIZE_ENVIRONMENT function is called during SO domain startup to create and initialize the CEEPIPI Language Environment pre-initialized environment for invokcation of C functions.

Output parameters

RESPONSE
is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOIS gate, INQUIRE function

The INQUIRE function is called by tasks that have been attached by the listener in response to a new TCP/IP connection. It provides TCP/IP and socket information about connection and the connected client.

Input parameters

[LISTEN_TOKEN]
is a token representing an opened tcpipservice.

[CLIENT_HOSTNAME]
is a buffer in which the full hostname of the client is returned to the caller.

[SERVER_HOSTNAME]
is a buffer in which the full hostname of the CICS region is returned to the caller.

[GENERIC_HOSTNAME]
is a buffer in which the full generic hostname of the CICS region, as known to the DNS in a connection optimization environment, is returned to the caller.

Output parameters

[CLIENT_IP_ADDRESS]
is the text representation of the IP address of the client.

[CLIENT_BIN_IP_ADDRESS]
is the 32 bit binary IP address of the client.

[SERVER_IP_ADDRESS]
is the text representation of the IP address of the CICS region.

[SERVER_BIN_IP_ADDRESS]
is the 32 bit binary IP address of the CICS region.

[LISTENER_PORT]
is the port number that the connection was received on.

[CLIENT_IP_ADDRESS_LEN]
is the length of the text representation of the client IP address.

[SERVER_IP_ADDRESS_LEN]
is the length of the text representation of the server IP address.

[CERTFICATE_USERID]
is the userid associated with the certificate that was used to authenticate a client if this is an SSL connection.

[SSLTYPE]
returns whether or not SSL is being used to secure this connection.

[URM_NAME]
is the name of the user-replaceable program specified on the tcpipservice definition for this connection.

[TSQ_PREFIX]
is the TS queue prefix specified on the tcpipservice definition for this connection.

[LISTENER_STATUS]
is the current status of the SO domain listener task.

[CONNECTIONS]
is either the number of connections for the service represented by the supplied LISTEN_TOKEN, or the total number of TCP/IP connections to all of of the currently active services.

[TCPIPSERVICE_NAME]
is the name of the service that attached the task, or the name associated with the supplied LISTEN_TOKEN.

[GROUP_NAME]
is the name of the dynamic DNS group that is registered with the MVS Work Load Manager for this service.

RESPONSE
is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOIS gate, SET function

The SET function is called to open, close or immediatly close the SO domain within a region. This is called in response to a SET TCPIP operator or SPI command.

Input parameters

[TCPIP_STATUS]
is either OPEN,CLOSED or IMMCLOSE.

Output parameters

RESPONSE
is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOIS gate, INQUIRE_STATISTICS function

The INQUIRE_STATISTICS function returns gathered statistics about an open tcpipservice.

Input parameters

LISTEN_TOKEN
is the token representing the open tcpipservice, returned from the SORD REGISTER function.

RESET
is a value indicating if the statistics should be reset.

Output parameters

[ATTACH_COUNT]
is the total number of tasks that have been attached to handle incoming connections.

[PEAK_CONNECTIONS]
is the high water mark for connections since that last reset.

[SEND_COUND] is the number of times TCP/IP send has been called.

[SEND_BYTES] is the number of bytes that have been sent to TCP/IP.

[RECV_COUNT] is the number of times TCP/IP receive has been called.

[RECV_BYTES] is the number of bytes received from TCP/IP.

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOIS gate, VERIFY function

The VERIFY function checks an IP address string in the form nnn.nnn.nnn.nnn for validity and returns the binary form.

Input parameters

SERVER_IP_ADDRESS
is a char string in the form nnn.nnn.nnn.nnn representing an IP address.

Output parameters

SERVER_BIN_IP_ADDRESS
is the 32 bit binary number of the IP address.

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOIS gate, EXPORT_CERTIFICATE_DATA function

The EXPORT_CERTIFICATE_DATA function saves a certificate in the sockets repository.

Input parameters

CERTIFICATE_INFORMATION
is a block representing the certificate.

Output parameters

REPOSITORY_TOKEN
is a token that represents the saves certificate data.

RESPONSE
is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOIS gate, IMPORT_CERTIFICATE_DATA function

The IMPORT_CERTIFICATE_DATA imports certificate data from the sockets repository.

Input parameters

[REPOSITORY_TOKEN]
a token representing a certificate exported to the repository.

CERTIFICATE_INFORMATION
is the block representing the certificate. The data is returned by the function.

Output parameters

CERTIFICATE_USERID
is the userid associated with the certificate.

RESPONSE
is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOIS gate, DELETE_CERTIFICATE_DATA function

The DELETE_CERTIFICATE_DATA deletes certificate data from the sockets repository.

Input parameters

REPOSITORY_TOKEN
a token representing a certificate exported to the repository.

Output parameters

RESPONSE
is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP LOCK_FAILURE

SOAD gate, ADD_REPLACE_TCPIPSERVICE function

The ADD_REPLACE_TCPIPSERVICE function is called at RDO time to install a tcpipservice definition. If the status is OPEN then the service is also opened using the SORD REGISTER function. A catalog entry is written to record the installed resource.

Input parameters

TCPIPSERVICE_NAME
is the name of the tcpipservice.

URM_NAME is the name of the user-replaceable program.

STATUS is either OPEN or CLOSED.

SSL is either YES, NO or CLIENTAUTH.

TRANSACTION is the tranid of the transaction to attach for each connection to this service.

BACKLOG is the TCP/IP listen backlog parameter.

PORTNUMBER is the port number to listen on.

[CERTIFICATE_LABEL]
is the name of the certificate from the keyfile to use to authenticate this service.

IPADDRESS is the IP address to bind this service to.

TSQPREFIX is the TS queue prefix to use for this service.

SOCKETCLOSE is the value of receive timeout for this service.

Output parameters

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOAD gate, DELETE_TCPIPSERVICE function

The DELETE_TCPIPSERVICE function is called at RDO time to remove an installed tcpipservice definition. If the status is OPEN then the tcpipservice is not removed. The catalog entry is removed for the discarded resource.

Input parameters

TCPIPSERVICE_NAME
is the name of the tcpipservice to remove.

Output parameters

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOTB gate, INQUIRE_TCPIPSERVICE function

The INQUIRE_TCPIPSERVICE function is called by CEMT and the SPI for an INQUIRE TCPIPSERICE function. It returns information about an installed tcpipservice.

Input parameters

TCPIPSERVICE_NAME
is the name of the tcpipservice to inquire upon.

Output parameters

[TRANSID] is the transaction ID associated with the service.
[URM] is the name of the user-replaceable program associated with the service.
[PORT] is the port number associated with the service.
[BACKLOG] is the backlog value associated with the service.
[CONNECTIONS]
is the current number of connections associated with the service.
[IPADDRESS] is the IP address that the service is bound to.
[TSQPREFIX] is the TS queue prefix associated with the service.
[SOCKETCLOSE]
is the receive timeout value associated with the service.
RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOTB gate, START_BROWSE function

The START_BROWSE function is called by CEMT and the SPI for an browsing tcpipservices.

Output parameters

BROWSE_TOKEN is a token representing the browse.
RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP LOCK_FAILURE

SOTB gate, GET_NEXT function

The GET_NEXT function is called by CEMT and the SPI for browsing tcpipservices. It returns information about an installed tcpipservice.

Input parameters

BROWSE_TOKEN is a token representing the browse.

Output parameters

TCPIPSERVICE_NAME
is the name of the tcpipserivce.

[TRANSID] is the transaction ID associated with the service.

[URM] is the name of the user-replaceable pogram associated with the service.

[PORT] is the port number associated with the service.

[BACKLOG] is the backlog value associated with the service.

[CONNECTIONS]
is the current number of connections associated with the service.

[IPADDRESS] is the IP address that the service is bound to.

[TSQPREFIX] is the TS queue prefix associated with the service.

[SOCKETCLOSE]
is the receive timeout value associated with the service.

[STATUS] is the current status of the service:

OPEN|OPENING|CLOSED|CLOSING|IMMCLOSING

[SSL] is the SSL setting for the service:

YES|NO|CLIAUTH

[CERTIFICATE_LABEL]
is the certificate label associated with the service.

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOTB gate, END_BROWSE function

The END_BROWSE function is called by CEMT and the SPI to end browsing tcpipservices.

Input parameters

BROWSE_TOKEN is a token representing the browse.

Output parameters

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOTB gate, SET_TCPIPSERVICE function

The SET_TCPIPSERVICE function is called by CEMT and the SPI to set tcpipservice parameters.

Input parameters

TCPIPSERVICE_NAME
is the name of the service to set.

[STATUS] is the status to set for the service:

OPEN|CLOSED|IMMCLOSED

[URM] is the name of the user-replaceable program.

[BACKLOG] is the value of the new backlog parameter. This can only be set if the service is closed.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOSE gate, INITIALIZE_SSL function

The INITIALIZE_SSL function is called at SO domain initialization. It creates the environment necessary to perform Secure Sockets Layer communication.

Output parameters

GSK_RETURN_CODE
is the return code from the System SSL component of OS/390 that CICS uses to perform SSL communications.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP CEEPIPI_ERROR GSK_ERROR

SOSE gate, SECURE_SOC_INIT function

The SECURE_SOC_INIT function is called when a new connection is established with the service and SSL is enabled. This function performs the SSL handshake to establish the security.

Output parameters

GSK_RETURN_CODE
is the return code from the System SSL component of OS/390 that CICS uses to perform SSL communications.

CERTIFICATE
is the certificate used by the client to authenticate itself.

CERTIFICATE_USERID
is the userid associated with the client certificate.

CIPHERS_SELECTED
represents the encryption cyphers that have been selected in negotiation with the client and server.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	GSK_INACTIVE,INSUFFICIENT_THREADS GETMAIN_FAILED,REPOSITORY_ERROR CONNECTION_CLOSED,CLIENT_ERROR
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP CEEPIPI_ERROR GSK_ERROR

SOSE gate, SECURE_SOC_READ function

The SECURE_SOC_READ function is called to read data on a secure connection.

Input parameters

RECEIVE_BUFFER
is the buffer to hold the received data.

Output parameters

GSK_RETURN_CODE
is the return code from the System SSL component of OS/390 that CICS uses to perform SSL communications.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP CEEPIPI_ERROR GSK_ERROR CONNECTION_CLOSED HANDSHAKE_ERROR

SOSE gate, SECURE_SOC_WRITE function

The SECURE_SOC_WRITE function is called to send data on a secure connection.

Input parameters

SEND_BUFFER is the buffer to holding the data to send.

Output parameters

GSK_RETURN_CODE
is the return code from the System SSL component of OS/390 that CICS uses to perform SSL communications.

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP CEEPIPI_ERROR GSK_ERROR CONNECTION_CLOSED HANDSHAKE_ERROR

SOSE gate, SECURE_SOC_CLOSE function

The SECURE_SOC_CLOSE function is called to close a secure connection.

Output parameters

RESPONSE is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP CEEPIPI_ERROR GSK_ERROR

SOSE gate, SECURE_SOC_RESET function

The SECURE_SOC_RESET function is called to reset a secure connection.

Output parameters

RESPONSE
is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP CEEPIPI_ERROR GSK_ERROR

SOSE gate, TERMINATE_SSL function

The TERMINATE_SSL function is to terminate all SSL operation in a region.

Output parameters

RESPONSE
is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP CEEPIPI_ERROR

SOSE gate, EXPORT_CERTIFICATE_DATA function

The EXPORT_CERTIFICATE_DATA function saves a certificate in the sockets repository.

Input parameters

CERTIFICATE_INFORMATION
is a block representing the certificate.

Output parameters

REPOSITORY_TOKEN
is a token that represents the saves certificate data.

RESPONSE
is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOSE gate, IMPORT_CERTIFICATE_DATA function

The IMPORT_CERTIFICATE_DATA imports certificate data from the sockets repository.

Input parameters

[REPOSITORY_TOKEN]
a token representing a certificate exported to the repository.

CERTIFICATE_INFORMATION
is the block representing the certificate. The data is returned by the function.

Output parameters

CERTIFICATE_USERID
is the userid associated with the certificate.

RESPONSE
is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

SOSE gate, DELETE_CERTIFICATE_DATA function

The DELETE_CERTIFICATE_DATA deletes certificate data from the sockets repository.

Input parameters

REPOSITORY_TOKEN
a token representing a certificate exported to the repository.

Output parameters

RESPONSE
is the domain└s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP LOCK_FAILURE

Socket domain's generic gates

Table 101 summarizes the socket domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 101. Socket domain's generic gates

Gate	Trace	Function	Format
SODM	SO 0101 SO 0102	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
SOST	SO 0A01 SO 0A02	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
SOXM	EM 0401 EM 0402	INQUIRE_DATA_LENGTH GET_DATA DESTROY_TOKEN	XMXM

For descriptions of these functions and their input and output parameters, refer to the Sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—Figure 99 on page 457

Format STST—"System programming command flows" on page 264

Format XMXM-Figure 76 on page 387

In initialization, quiesce, and termination processing, the socket domain performs only internal routines.

Modules

Module	Function
DFHSODM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHSOCK	Handles the following requests: LISTEN SEND RECEIVE CLOSE SEND_SSL_DATA RECV_SSL_DATA
DFHSORD	Handles the following requests: REGISTER DEREGISTER IMMCLOSE
DFHSOIS	Handles the following requests: INITIALIZE_ENVIRONMENT INQUIRE SET_PARAMETERS INQUIRE_STATISTICS VERIFY EXPORT_CERTIFICATE_DATA IMPORT_CERTIFICATE_DATA DELETE_CERTIFICATE_DATA

Module	Function
DFHSOAD	Handles the following requests: ADD_REPLACE_TCPIPSERVICE DELETE_TCPIPSERVICE
DFHSOTB	Handles the following requests: INQUIRE_TCPIPSERVICE START_BROWSE GET_NEXT END_BROWSE SET_TCPIPSERVICE
DFHSOSE	Handles the following requests: INITIALIZE_SSL SECURE_SOC_INIT SECURE_SOC_READ SECURE_SOC_WRITE SECURE_SOC_CLOSE SECURE_SOC_RESET TERMINATE_SSL EXPORT_CERTIFICATE_DATA IMPORT_CERTIFICATE_DATA DELETE_CERTIFICATE_DATA
DFHSODUF	Formats the SO domain control blocks
DFHSOTRI	Interprets SO domain trace entries

Exits

No global user exit points are provided in this domain.

Chapter 107. Statistics domain (ST)

The statistics domain controls the collection of resource statistics for a CICS system (the monitoring domain collects task statistics). The statistics domain collects data at user-specified intervals, at system quiesce or logical end of day, and when requested by the user, and writes it to the statistics data sets in SMF format. This can subsequently be used by the statistics offline utility to produce formatted reports.

Statistics domain's specific gate

Table 102 summarizes the statistics domain's specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 102. Statistics domain's specific gate

Gate	Trace	Function	XPI
STST	ST 0003	INQ_STATISTICS_OPTIONS	NO
	ST 0004	SET_STATISTICS_OPTIONS	NO
		REQUEST_STATISTICS	NO
		RECORD_STATISTICS	NO
		STATISTICS_COLLECTION	NO
		DISABLE_STATISTICS	NO

STST gate, INQ_STATISTICS_OPTIONS function

The INQ_STATISTICS_OPTIONS function of the STST gate is used to return information associated with the statistics domain options.

Input parameters

None.

Output parameters

COLLECT
indicates whether interval statistics are being collected (and their counts reset). It can have either of these values:

YES|NO

INTERVAL
is the interval at which statistics are being collected if COLLECT is YES.

EOD_TIME_OF_DAY
is the time of day at which end-of-day statistics are collected.

NEXT_COLLECTION_TIME
is the time of the next collection of statistics. If COLLECT is YES, it is the earlier of the next interval collection time and the logical end-of-day time; if COLLECT is NO, it is the logical end-of-day time.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|INVALID|KERNERROR|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

STST gate, SET_STATISTICS_OPTIONS function

The SET_STATISTICS_OPTIONS function of the STST gate is used to set statistics options.

Input parameters

[COLLECT]
indicates whether interval statistics are to be collected (and their counts reset). It can have either of these values:

YES|NO

[INTERVAL] is the interval at which statistics are to be collected if COLLECT is YES.

[EOD_TIME_OF_DAY]
is the time of day at which end-of-day statistics are to be collected.

[COLLECT_UPDATE_ACTION]
is the action to be taken when changing the COLLECT option value from NO to YES, or from YES to NO. It can have any one of these values:

NOACTION|RESETNOW|RECORDNOW|RECORD_RESETNOW

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|INVALID|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	COLL_ACTION_NO_UPDATE
INVALID	INVALID_COLLECT INVALID_INTERVAL INVALID_EOD_TIME_OF_DAY INV_COLL_UPDATE_ACTION

STST gate, REQUEST_STATISTICS function

The REQUEST_STATISTICS function of the STST gate is used to request a collection of statistics.

Input parameters

[DOMAIN_TOKEN]
identifies the domain from which the statistics are to be collected.

[RESOURCE_TYPE]
indicates the resource in the AP domain on which statistics are to be collected.

REQUEST_TOKEN
uniquely identifies the collection of statistics requested by the caller.

RESET indicates whether certain statistics fields are to be reset.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|INVALID|EXCEPTION|PURGED|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TYPE_NOT_FOUND NOT_AVAILABLE INCOMPLETE_DATA
INVALID	INVALID_RESET

STST gate, RECORD_STATISTICS function

The RECORD_STATISTICS function of the STST gate is used to record statistics.

Input parameters

STATISTICS_DATA
: specifies the address and length of data requested.

STATISTICS_TYPE
: indicates the type of statistics collection, either a normal collection or unsolicited. It can have either of these values:

  COLLECTION|USS

Output parameters

RESPONSE
: is the domain's response to the call. It can have any of these values:

  OK|INVALID|KERNERROR|DISASTER

[REASON]
: is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_DATA_FORMAT

STST gate, STATISTICS_COLLECTION function

The STATISTICS_COLLECTION function of the STST gate is used to initiate a collection of statistics.

Input parameters

RESET
: indicates whether certain statistics fields are to be reset.

DATA
: indicates whether the domain being called is requested to return its statistics to the caller.

END_OF_DAY
: indicates whether all statistics fields are to be reset.

COLLECTION_TYPE
: indicates whether this is an interval collection or end-of-day collection of statistics. It can have either of these values:

  INT|EOD

[SYSTEM_TERMINATING]
: indicates whether this is the last collection for the CICS run. It can have either of these values:

  YES|NO

  YES is used for the end-of-day collection that is taken when CICS is shut down.

Output parameters

RESPONSE
: is the domain's response to the call. It can have any of these values:

  OK|INVALID|KERNERROR|DISASTER

[REASON]
: is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

STST gate, DISABLE_STATISTICS function

The DISABLE_STATISTICS function of the STST gate is used to disable statistics interval collections.

Input parameters

None.

Output parameters

RESPONSE
: is the domain's response to the call. It can have any of these values:

OK|INVALID|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

Statistics domain's generic gates

Table 103 summarizes the statistics domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 103. Statistics domain's generic gates

Gate	Trace	Function	Format
DMDM	ST 0001 ST 0002	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
TISR	ST 0005 ST 0006	NOTIFY	TISR

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669

Format TISR—"Timer domain's generic format" on page 1203

In initialization processing, the statistics domain sets the initial statistics options:
- Collecting interval
- Logical end of day
- Collecting status.

For a cold start, the collecting interval defaults to 3 hours, the logical end of day defaults to midnight, and the collecting status defaults to ON; for any other type of start, the information comes from the global catalog.

In quiesce processing, the statistics domain collects and records statistics from all other domains.

In termination processing, the statistics domain collects and records end-of-day statistics.

Statistics domain's generic format

Table 104 summarizes the generic format owned by the statistics domain and shows the functions performed on the calls.

Table 104. Generic format owned by statistics domain

Format	Calling module	Function
STST	DFHSTST DFHEIQMS	COLLECT_STATISTICS COLLECT_RESOURCE_STATS

In the descriptions of the format that follows, the “input” parameters are input not to statistics domain, but to the domain being called by the statistics domain. Similarly, the “output” parameters are output by the domain that was called by the statistics domain, in response to the call.

STST format, COLLECT_STATISTICS function

The COLLECT_STATISTICS function of the STST format is used by the statistics domain to ask a domain to collect its statistics.

Input parameters

DATA
indicates whether the domain being called is requested to return its statistics to the caller. It can have either of these values:

YES|NO

END_OF_DAY
indicates whether all statistics fields are to be reset. It can have either of these values:

YES|NO

RESET
indicates whether certain statistics fields are to be reset. It can have either of these values:

YES|NO

RESET_TIME
is the time of day to be used as the time at which the statistics fields were last reset.

[RESOURCE_TYPE]
indicates the resource in the AP domain on which statistics are to be collected.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|PURGED|KERNERROR|DISASTER

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TYPE_NOT_FOUND NOT_AVAILABLE INCOMPLETE_DATA

STST format, COLLECT_RESOURCE_STATS function

The COLLECT_RESOURCE_STATS function of the STST format is used by the EXEC API to ask a domain to collect its monitoring data collection information.

Input parameters

[RESOURCE_TYPE]
is the type of resource on which statistics are required.

[RESOURCE_ID]
specifies the address and length of the resource identifier.

[RESOURCE_ID_2]
specifies the address and length of the resource identifier.

[RESOURCE_ID_3]
specifies the address and length of the resource identifier.

[LONG_RESOURCE_ID_DATA]
specifies the address and length of the resource identifier.

[RESID_TOKEN]
a token representing the resource id required.

RESOURCE_STATISTICS_DATA
specifies the address and length of the area into which the requested statistics are to be placed.

Output parameters

[LAST_RESET_TIME]
indicates the time at which the statistics fields were last reset.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|PURGED|KERNERROR|DISASTER

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TYPE_NOT_FOUND ID_NOT_FOUND NOT_AVAILABLE

Modules

Module	Function
DFHSTDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHSTDUF	Formats the ST domain control blocks in a CICS system dump
DFHSTST	Handles the following requests: INQ_STATISTICS_OPTIONS RECORD_STATISTICS REQUEST_STATISTICS SET_STATISTICS_OPTIONS STATISTICS_COLLECTION DISABLE_STATISTICS
DFHSTTI	Handles the NOTIFY request
DFHSTTRI	Interprets ST domain trace entries
DFHSTUE	Provides a SET_EXIT_STATUS routine to enable or disable a user exit.

Exits

There is one global user exit point in the statistics domain: XSTOUT. See the *CICS Customization Guide* for further information.

Trace

The point IDs for the statistics domain are of the form ST xxxx; the corresponding trace levels are ST 1, ST 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 108. Timer domain (TI)

The timer domain provides interval timing and alarm clock services for CICS domains. These are processes that cause an action to occur at some predetermined future time. This service (called "notifying") can be performed after a specific interval, at periodic intervals, at a specified time of day, or at a specific time of day every day.

The timer domain also provides date and time provision and conversion functions. This includes the facility to synchronize the CICS local time with the operating clock when the system operator has adjusted the time zone.

Timer domain's specific gate

Table 105 summarizes the timer domain's specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 105. Timer domain's specific gate

Gate	Trace	Function	XPI
TISR	TI 0100	REQUEST_NOTIFY_INTERVAL	NO
	TI 0101	REQUEST_NOTIFY_TIME_OF_DAY	NO
		CANCEL	NO
		INQUIRE_EXPIRATION_TOKEN	NO

TISR gate, REQUEST_NOTIFY_INTERVAL function

The REQUEST_NOTIFY_INTERVAL function of the TISR gate is used to request the timer domain to notify the calling domain after a specified real interval of time. The calling domain can request a NOTIFY on a one-off basis or periodically, and can specify the type of NOTIFY to be expected.

Input parameters

DOMAIN_TOKEN
is a token that is to be passed as a parameter on the NOTIFY call.

STCK_INTERVAL
specifies an interval as a doubleword binary interval in stored clock (STCK) format, where bit 51 of the doubleword represents 1 microsecond.

PERIODIC_NOTIFY
specifies whether the requested NOTIFY is to be repeated at the specified interval until canceled (YES), or is to be just a one-off NOTIFY (NO). It can have either of these values:

YES|NO

NOTIFY_TYPE
specifies whether the attached task or the timer task is to be used to notify the calling domain after the specified interval of time. It can have either of these values:

ATTACHED_TASK|TIMER_TASK

[ATTACH_PRIORITY]
defines the priority, in the range 0 through 255, at which the requested NOTIFY task is to be attached.

[ATTACH_TASK_TIMEOUT]
defines the value, in seconds, of a wait in the attached task after which the dispatcher causes a time-out.

[ATTACH_MODE]
is the optional TCB mode in which the attached NOTIFY task is to run.

[ORIGIN_DATE]
defines the date from which the timer domain is to start the interval timing for this request. This parameter is mandatory if ORIGIN_TIME has been specified. It holds the origin date as MMDDYYYY.

[ORIGIN_TIME]
defines the local time of day from which the timer domain is to start the interval timing for this request. The value in decimal digits is specified in the form HHMMSS:

HH Hours in the range 00 through 23
MM Minutes in the range 00 through 59
SS Seconds in the range 00 through 59.

ORIGIN_TIME defaults to the current time.

Output parameters

TIMER_TOKEN is the token that is returned by the timer domain. The timer token may be used to cancel the NOTIFY request.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|INVALID|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. It has this value:

INVALID_INTERVAL

TISR gate, REQUEST_NOTIFY_TIME_OF_DAY function

The REQUEST_NOTIFY_TIME_OF_DAY function of the TISR gate is used to inform the timer domain that an alarm call is required from the timer domain (that is, a NOTIFY) at the specified time of day. The calling domain can request a NOTIFY on a one-off basis or daily, and the type of NOTIFY to be expected.

Input parameters

DOMAIN_TOKEN is the token that is to be passed as a parameter on the NOTIFY call.

REQUESTED_TIME
is the time of day at which the NOTIFY function is to be invoked. The value is specified in the form HHMMSS.

PERIODIC_NOTIFY
specifies whether the requested NOTIFY is to be repeated every day at the requested time (YES), or is to be just a one-off NOTIFY (NO). It can have either of these values:

YES|NO

NOTIFY_TYPE specifies whether the attached task or the timer task is to be used to notify the calling domain after the specified interval of time. It can have either of these values:

ATTACHED_TASK|TIMER_TASK

[ATTACH_PRIORITY]
defines the priority, in the range 0 through 255, at which the requested NOTIFY task is to be attached.

[ATTACH_TASK_TIMEOUT]
defines the value, in seconds, of a wait in the attached task after which the dispatcher causes a time-out.

[ATTACH_MODE]
is the optional TCB mode in which the attached NOTIFY task is to run.

Output parameters

TIMER_TOKEN is the token that is returned by the timer domain.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|INVALID|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. It has this value:

TOO_LATE

TISR gate, CANCEL function

The CANCEL function of the TISR gate is used to cancel a timer request that has already been initiated by one of these functions:

REQUEST_NOTIFY_INTERVAL
REQUEST_NOTIFY_TIME_OF_DAY

Input parameters

TIMER_TOKEN
is the token that was returned when the timer request was made.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|INVALID|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	REQUEST_NOT_FOUND TOO_LATE

TISR gate, INQUIRE_EXPIRATION_TOKEN function

The INQUIRE_EXPIRATION_TOKEN function of the TISR gate is used by the dispatcher domain during its initialization.

Input parameters

None.

Output parameters

EXPIRATION_TOKEN
is a token used during initialization of the dispatcher domain.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|INVALID|EXCEPTION|DISASTER|KERNERROR|PURGED

Timer domain's generic gate

Table 106 summarizes the timer domain's generic gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 106. Timer domain's generic gate

Gate	Trace	Function	Format
DMDM	TI 0001 TI 0002	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

You can find descriptions of these functions and their input and output parameters in the section dealing with the corresponding generic format, in format DMDM under "Domain manager domain's generic formats" on page 669.

In initialization and quiesce processing, the timer domain performs only internal routines.

The timer domain does no termination processing.

Timer domain's generic format

Table 107 describes the generic format owned by the timer domain and shows the function performed on the calls.

Table 107. Generic format owned by the timer domain

Format	Calling module	Function
TISR	DFHTISR	NOTIFY

In the descriptions of the formats that follow, the “input” parameters are input not to timer domain, but to the domain being called by the timer. Similarly, the “output” parameters are output by the domain that was called by timer domain, in response to the call.

TISR format, NOTIFY function

The NOTIFY function of the TISR format is used by the timer domain itself to notify a domain after its requested interval or time has expired.

Input parameters

DOMAIN_TOKEN is a token that is to be passed as a parameter on the NOTIFY call.

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|INVALID|EXCEPTION|DISASTER|KERNERROR|PURGED

Modules

Module	Function
DFHTIDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHTIDUF	Formats the timer domain’s control blocks
DFHTISR	Handles the following requests: REQUEST_NOTIFY_INTERVAL REQUEST_NOTIFY_TIME_OF_DAY CANCEL INQUIRE_EXPIRATION_TOKEN
DFHTITRI	Interprets timer domain trace entries

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the timer domain are of the form TI xxxx; the corresponding trace levels are TI 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 109. Trace domain (TR)

The trace domain is used by CICS system code and user application programs to record details of the sequence of events occurring in the system. The basic unit of information created for this purpose is called a **trace entry**. The trace domain can put trace entries to any combination of three possible destinations:

INTERNAL trace
: a wraparound table in main storage in the CICS address space

AUXILIARY trace
: a pair of CICS-controlled BSAM data sets used alternately

Generalized trace facility (GTF) trace
: the user-defined destination for MVS GTF records.

Design overview

The trace domain consists of a set of modules that are used to record and manage trace information about internal, auxiliary, and GTF trace. The services of the trace domain are requested by making domain calls, described in “Domain calls” on page 1206. The modules that handle these domain calls are DFHTRDM, DFHTRPT, and DFHTRSR.

Certain sub-functions of the trace domain are required by more than one of these modules. These sub-functions are packaged together in the DFHTRSU module, and are invoked by domain subroutine calls, described in “Subroutine calls” on page 1209.

All processing directly related to the auxiliary trace data sets is carried out by the DFHTRAO module. DFHTRAO is loaded below the 16MB line so that it can run in 24-bit mode when calling BSAM and referencing the auxiliary trace data set data control block (DCB). The DFHTRAO functions are described in “DFHTRAO functions” on page 1210.

TRACE_PUT handling

For performance reasons, it is important to minimize the path length of a request to write a trace entry. This is achieved for most TRACE_PUT requests by handling them in module DFHTRPX, which runs as a subroutine of the domain that is requesting the trace.

DFHTRPX runs in a very restricted environment. It has no working storage and can make no calls out. Nevertheless, it can still handle the majority of TRACE_PUT requests. When DFHTRPX cannot handle a request, it passes control to the TRPT gate of the trace domain for module DFHTRPT to process the request.

DFHTRPX passes control to DFHTRPT in the following situations:

- CICS tracing to GTF is active.
- Transaction dump processing currently holds the trace lock while copying parts of the trace table to a local buffer.
- DFHTRSR currently holds the trace lock while processing the SET_INTERNAL_TABLE_SIZE function.
- CICS auxiliary trace is active and the requested entry does not fit in the current block, that is, a block write is required.
- The amount of data passed for tracing is larger than the trace domain limit (overlength entry).
- DFHTRPX’s recovery routine has been driven, probably because of a program check while moving data into the internal trace table.
- The FE global trap/trace exit (DFHTRAP) is active.

Locking

The trace domain handles TRACE_PUT requests from many MVS task control blocks (TCBs), and so requires a locking mechanism to prevent overlapping or simultaneous access to its control blocks. This is an MVS TCB lock and is provided by the LOCK and UNLOCK functions of the DFHKERN macro.

DFHTRPX does not acquire the trace lock. It uses “compare double and swap” (CDS) logic to serialize the allocation of space for trace entries in the internal trace table.

Selectivity

The overall trace master flag is logically a combination of the flags controlling internal, auxiliary, and GTF trace. It is owned by the trace domain, and both the kernel and the common system area (CSA) have their own copies that are kept up-to-date by calls from the trace domain.

The user trace master flag is owned by the AP domain. The system trace master flag and the standard and special component trace flags are owned by the kernel. None of these flags is referenced by the trace domain.

Domain calls

This section lists the process flows for the domain calls used for the trace domain services.

DMDM gate, PRE_INITIALIZE function

1. Issue an MVS GETMAIN for the trace domain anchor block (TRA) and initialize it.
2. Acquire startup information from the parameter manager (PA) domain and set it in the TRA. The relevant startup parameters are INTTR, TRTABSZ, AUXTR, AUXTRSW, and GTFTR.
3. Call TRSU SET_UP_INTERNAL_TABLE to get and initialize the internal trace table.
4. Call TRSU GET_GTF_BUFFER to initialize CICS tracing to GTF.
5. Issue the KEDD ADD_GATE call for the DFHTRPT gate to inform the kernel that the trace domain is available.
6. If internal trace or GTF trace is started, turn on the trace master flags in the TRA, the kernel, and the CSA.

DMDM gate, INITIALIZE_DOMAIN function

1. If required, call TRSR ACTIVATE_TRAP to active the FE global trap/trace exit, DFHTRAP.
2. If required, call TRSR START_AUXILIARY_TRACE to start auxiliary trace on DFHAUXT.

DMDM gate, QUIESCE_DOMAIN function

Do nothing.

DMDM gate, TERMINATE_DOMAIN function

If auxiliary trace is active, call TRSR STOP_AUXILIARY_TRACE.

KETI gate, NOTIFY_RESET function

Call KETI CONVERT_TO_STCK_FORMAT to get the new STCK value for the last local midnight, and store this in the TRA.

TRPT gate, TRACE_PUT function

1. Acquire the trace lock.
2. Calculate the length of the required entry.
3. If the entry does not fit in the current trace block (TRBL) and auxiliary trace is active, call TRSU WRITE_AUX_BUFFER.
4. Use “compare double and swap” (CDS) to update pointer and available length for next entry in the TRA.

5. Build the entry in allocated space.
6. If GTF trace is required, issue the GTRACE macro to write an entry to GTF, and if the entry is more than 256 bytes, split it into multiple entries.
7. If the FE global trap/trace exit, DFHTRAP, has been activated as a result of using the CSFE DEBUG transaction, or specifying the TRAP=ON system initialization parameter, invoke the exit. See the *CICS Problem Determination Guide* for details of DFHTRAP.
8. Release the trace lock.

TRSR gate, SET_INTERNAL_TABLE_SIZE function

1. If the call is from the parameter manager (during initialization), set the required size in the TRA and return.
2. Acquire the trace lock.
3. If auxiliary trace is active, call TRSU WRITE_AUX_BUFFER to write the current TRBL.
4. If the new table size is smaller, free part of the old table and reset chaining and pointers.
5. If a larger table is required, free all but 16KB (KB equals 1024 bytes) of the old table. Call TRSU SET_UP_INTERNAL_TABLE. If this completes correctly, free the 16KB that was kept back. If it does not work, make the 16KB piece the new table.
6. Release the trace lock.

TRSR gate, START_INTERNAL_TRACE function

1. Set the required status in the TRA.
2. If the call is from the parameter manager (during initialization), return.
3. If required, change the kernel and CSA copies of the trace master flag.

TRSR gate, STOP_INTERNAL_TRACE function

1. Set the required status in the TRA.
2. If the call is from the parameter manager (during initialization), return.
3. If required, change the kernel and CSA copies of the trace master flag.

TRSR gate, INQUIRE_INTERNAL_TRACE function

Get the internal status and internal table size from the TRA.

TRSR gate, START_AUXILIARY_TRACE function

1. If the call is from the parameter manager (during initialization), set the status in the TRA and return.
2. If already started, return immediately.
3. If auxiliary trace is currently stopped (rather than paused):
   a. Issue an MVS GETMAIN for an auxiliary trace buffer, DCB, and DECB storage.
   b. Issue LDLD ACQUIRE_PROGRAM for DFHTRAO.
   c. Call DFHTRAO to OPEN the auxiliary trace data set.
4. Acquire the trace lock.
5. Skip the current TRBL pointer in the TRA to the next TRBL to avoid entries from before start appearing in the auxiliary trace.
6. Release the trace lock.
7. Set the auxiliary trace status in the TRA to started.
8. If required, change the kernel and CSA copies of the trace master flag.

TRSR gate, STOP_AUXILIARY_TRACE function

1. If the call is from the parameter manager (during initialization), set the status in the TRA and return.
2. If already stopped, return immediately.
3. Acquire the trace lock.

4. If auxiliary trace is started (rather than paused), call TRSU WRITE_AUX_BUFFER to output the current TRBL to the auxiliary trace data set, and move the current TRBL pointer in the TRA to the next TRBL.
5. Call TRSU WRITE_AUX_BUFFER to write an end-of-file indication on the auxiliary trace data set.
6. Call DFHTRAO to ensure (CHECK) that output is complete.
7. Call DFHTRAO to CLOSE the auxiliary trace data set.
8. Call TRSU TERMINATE_AUXILIARY_TRACE.
9. Release the trace lock.
10. Issue LDLD RELEASE_PROGRAM for DFHTRAO.

TRSR gate, PAUSE_AUXILIARY_TRACE function

1. If auxiliary trace is stopped, return with error.
2. If auxiliary trace is paused, return 'OK'.
3. Acquire the trace lock.
4. Call TRSU WRITE_AUX_BUFFER to output the current TRBL to the auxiliary trace data set, and move the current TRBL pointer in the TRA to the next TRBL.
5. Release the trace lock.
6. Change the kernel and CSA copies of the trace master flag if required.

TRSR gate, SET_AUX_TRACE_AUTOSWITCH function

Set the new autoswitch status in the TRA.

TRSR gate, SWITCH_AUXILIARY_EXTENTS function

1. If auxiliary trace is started or paused:
   a. Acquire the trace lock.
   b. Call TRSU WRITE_AUX_BUFFER to write an end-of-file indication on the auxiliary trace data set.
   c. Call DFHTRAO to ensure (CHECK) that output is complete.
   d. Call DFHTRAO to close the auxiliary trace data set.
2. Change the name of the current extent in the TRA from DFHAUXT to DFHBUXT or from DFHBUXT to DFHAUXT.
3. If auxiliary trace is started or paused:
   a. Call DFHTRAO to OPEN the auxiliary trace data set.
   b. Release the trace lock.

TRSR gate, INQUIRE_AUXILIARY_TRACE function

Get the auxiliary trace status, current extent name, and autoswitch status from the TRA.

TRSR gate, START_GTF_TRACE function

1. If the call is from the parameter manager (during initialization), set the required status in the TRA and return.
2. If already started, return immediately.
3. Call TRSU GET_GTF_BUFFER.
4. Set the status in the TRA to started.
5. If required, change the kernel and CSA copies of the trace master flag.

TRSR gate, STOP_GTF_TRACE function

1. Set the status in the TRA to stopped.
2. If the call is from the parameter manager (during initialization), return.
3. If required, change the kernel and CSA copies of the trace master flag.
4. If the GTF buffer is present:

   a. Acquire the trace lock.
   b. Issue an MVS FREEMAIN for the GTF buffer.
   c. Release the trace lock.

TRSR gate, INQUIRE_GTF_TRACE function

Get the GTF status from the TRA.

TRSR gate, ACTIVATE_TRAP function

1. If the call is from the parameter manager (during initialization), set the required status in the TRA and return.
2. If the trap is already active, check whether it is marked unusable because a program check occurred while the trap was in control:
   a. If the trap is unusable, return with error.
   b. If the trap is usable, set the required status in the TRA and return.
3. Issue LDLD ACQUIRE_PROGRAM for DFHTRAP.
4. Issue an MVS GETMAIN for the DFHTRAP work area (TRGTW).
5. Acquire the trace lock.
6. Check whether another task has activated the trap:
   a. If the trap is not active, update the trap status in the TRA, release the trace lock, and return.
   b. If the trap has been activated by another task, release the trace lock, issue LDLD RELEASE_PROGRAM for DFHTRAP, issue an MVS FREEMAIN for the DFHTRAP work area, and return.

TRSR gate, DEACTIVATE_TRAP function

1. If the call is from the parameter manager (during initialization), set the required status in the TRA and return.
2. If the trap is not active, return.
3. Acquire the trace lock.
4. Update the trap status in the TRA.
5. Release the trace lock.
6. Issue LDLD RELEASE_PROGRAM for DFHTRAP.
7. Issue an MVS FREEMAIN for the DFHTRAP work area.

Subroutine calls

This section lists the process flows for the domain subroutine calls used for the trace domain sub-functions.

TRSU format, WRITE_AUX_BUFFER function

1. If output to the auxiliary trace data set is pending, call DFHTRAO with a CHECK request to allow output to complete.
2. If there was no output pending or the output completed successfully, call DFHTRAO to write the current TRBL, and return.

If an ‘end of extent’ was encountered on the BSAM CHECK:

3. Call DFHTRAO to close the auxiliary trace data set.
4. If autoswitch is not required:
   a. Issue an MVS FREEMAIN for the auxiliary trace buffer, DCB, and DECB.
   b. Set auxiliary trace status in the TRA to stopped.
   c. Change the kernel and CSA copies of the trace master flag if required.
   d. Return.

5. If autoswitch next is specified, change to autoswitch off.
6. Change the name of the current extent in the TRA from DFHAUXT to DFHBUXT or from DFHBUXT to DFHAUXT.
7. Call DFHTRAO to OPEN the auxiliary trace data set.
8. Call DFHTRAO with a WRITE request to rewrite the block that caused the end-of-extent.
9. Go back to the top of this function's processing to issue the write that was originally requested in this call.

TRSU format, TERMINATE_AUXILIARY_TRACE function

1. Issue an MVS FREEMAIN for the auxiliary trace buffer, DCB, and DECB.
2. Set the auxiliary trace status in the TRA to stopped.
3. If required, change the kernel and CSA copies of the trace master flag.

TRSU format, GET_GTF_BUFFER function

1. Issue an MVS GETMAIN for the GTF buffer.
2. Save the address in the TRA.

TRSU format, SET_UP_INTERNAL_TABLE function

1. Issue an MVS V-type GETMAIN for the required size.
2. Initialize all TRBL headers within the acquired area.

DFHTRAO functions

This section lists the process flows for the DFHTRAO functions for auxiliary trace data sets.

DFHTRAO, OPEN function

1. If the DCB indicates already open, return 'OK'.
2. Issue the BSAM OPEN macro.

DFHTRAO, CLOSE function

1. If the DCB indicates already closed, return 'OK'.
2. Issue the BSAM CLOSE macro.

DFHTRAO, CHECK function

1. Issue the BSAM CHECK macro.
2. If an end-of-extent is caused by the write for which this CHECK is issued, the DCB ABEND exit is driven and causes DFHTRAO to return an end-of-extent indication to the caller.
3. Clear output pending status in TRA.

DFHTRAO, WRITE function

1. Move the specified TRBL to the auxiliary trace buffer.
2. Issue the BSAM WRITE macro.
3. Set output pending status in the TRA.

Trace domain's specific gates

Table 108 summarizes the trace domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 108. Trace domain's specific gates

Gate	Trace	Function	XPI
TRPT	None	TRACE_PUT	YES

Table 108. Trace domain's specific gates (continued)

Gate	Trace	Function	XPI
TRSR	TR 0201	SET_INTERNAL_TABLE_SIZE	NO
	TR 0202	START_INTERNAL_TRACE	NO
		STOP_INTERNAL_TRACE	NO
		INQUIRE_INTERNAL_TRACE	NO
		START_AUXILIARY_TRACE	NO
		STOP_AUXILIARY_TRACE	NO
		PAUSE_AUXILIARY_TRACE	NO
		SET_AUX_TRACE_AUTOSWITCH	NO
		SWITCH_AUXILIARY_EXTENTS	NO
		INQUIRE_AUXILIARY_TRACE	NO
		START_GTF_TRACE	NO
		STOP_GTF_TRACE	NO
		INQUIRE_GTF_TRACE	NO
		ACTIVATE_TRAP	NO
		DEACTIVATE_TRAP	NO

TRPT gate, TRACE_PUT function

This function is invoked to write a trace entry to the active trace destinations.

Input parameters

POINT_ID
is a number, unique within the calling domain, that identifies the trace entries made from this call.

[DATA1] through [DATA7]
are BLOCK descriptions of up to seven areas to be included in the data section of the trace entry. They appear in numerical order in the entry, each preceded by a 2-byte length field.

The maximum total length of data that can be traced in one call is as described below:

```
Length of trace table block              4096
less length of trace table block header  -   24
less length of trace entry header        -   32
                                         ------
Maximum space for data + length fields    4040
For each DATA field specified, 2 bytes must be
subtracted to allow for the length field.
Maximum space for actual data  =  4040 - (2 * n)
where 'n' is the number of DATA fields specified.
```

[RETURN_ADDR]
is used by DFHTRP to give a return address in the trace entry from the calling module rather than in DFHTRP.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

Note: No response is returned when the TRACE_PUT request is handled by module DFHTRPX without involving the trace domain.

TRSR gate, SET_INTERNAL_TABLE_SIZE function

The SET_INTERNAL_TABLE_SIZE function of the TRSR gate is used to change the size of the internal trace table during a CICS run.

Input parameters

TABLE_SIZE
is the required table size, specified as a number of KB (KB equals 1024 bytes). This is rounded up to the nearest multiple of 4KB. The lower limit is 16KB. The upper limit is set only by the amount of storage available. If the table is being made larger, the existing table is freed and a variable MVS GETMAIN issued for the required size. The actual

length of the new table can be determined by issuing an INQUIRE_INTERNAL_TRACE command. If the table is being made smaller, part of the existing table is freed.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_TABLE_SIZE NO_SPACE

Note: INVALID_TABLE_SIZE indicates that the value of TABLE_SIZE is less than 16KB.

NO_SPACE indicates that the variable GETMAIN for the new trace table failed to obtain even the minimum trace table size. In this situation, the trace domain retains an amount equal to the minimum table size from the old table to use.

TRSR gate, START_INTERNAL_TRACE function

The START_INTERNAL_TRACE function of the TRSR gate is used to activate tracing to the internal trace table.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

TRSR gate, STOP_INTERNAL_TRACE function

The STOP_INTERNAL_TRACE function of the TRSR gate is used to deactivate tracing to the internal trace table.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

TRSR gate, INQUIRE_INTERNAL_TRACE function

The INQUIRE_INTERNAL_TRACE function of the TRSR gate is used to return the status of the internal trace and the current size of the internal trace table.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

TABLE_SIZE
is the size of the current internal trace table in KB (KB equals 1024 bytes).

INTERNAL_STATUS
indicates whether internal trace is active (STARTED) or inactive (STOPPED).

TRSR gate, START_AUXILIARY_TRACE function

The START_AUXILIARY_TRACE function of the TRSR gate is used to open the current auxiliary trace extent (if it is closed) and start tracing to it.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CANT_GET_AUX_BUFFER DFHTRAO_NOT_AVAILABLE OPEN_FAILED
Note: CANT_GET_AUX_BUFFER indicates that MVS had insufficient free storage to satisfy the request for a buffer below the 16MB line. DFHTRAO_NOT_AVAILABLE indicates that the request to the CICS loader to acquire the auxiliary trace output program, DFHTRAO, has failed. OPEN_FAILED indicates that the MVS open of the auxiliary trace data set has failed.	

TRSR gate, STOP_AUXILIARY_TRACE function

The STOP_AUXILIARY_TRACE function of the TRSR gate is used to stop auxiliary tracing and close the currently active auxiliary trace extent.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

TRSR gate, PAUSE_AUXILIARY_TRACE function

The PAUSE_AUXILIARY_TRACE function of the TRSR gate is used to stop auxiliary tracing without closing the currently active extent.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:

AUX_TRACE_STOPPED

meaning that the pause is allowed only if auxiliary trace is not stopped.

TRSR gate, SET_AUX_TRACE_AUTOSWITCH function

The SET_AUX_TRACE_AUTOSWITCH function of the TRSR gate is used to allow the autoswitch facility for the CICS auxiliary trace data set to be enabled or disabled.

Input parameters

AUTOSWITCH_STATUS
Indicates whether or not an automatic switch to the inactive CICS auxiliary extent is to occur once only when the current extent fills up, or that such automatic switching should occur “continuously” whenever the current extent fills up. It can have any one of these values:

OFF|ONCE|CONTINUOUS

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:

INVALID_AUTOSWITCH_STATUS

meaning that an incorrect value was passed for AUTOSWITCH_STATUS.

TRSR gate, SWITCH_AUXILIARY_EXTENTS function

The SWITCH_AUXILIARY_EXTENTS function of the TRSR gate allows switching from one auxiliary trace extent to the other.

Input parameters

None.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:

OPEN_FAILED

meaning that the attempt to open the new auxiliary extent failed.

TRSR gate, INQUIRE_AUXILIARY_TRACE function

The INQUIRE_AUXILIARY_TRACE function of the TRSR gate is used to return the current state of the auxiliary trace.

Input parameters

None.

Output parameters

AUXILIARY_STATUS
Indicates the current status of auxiliary trace. It can have any one of these values:

STARTED|STOPPED|PAUSED

EXTENT
indicates the currently active CICS auxiliary trace extent; that is, the extent that is already in use or is used if CICS auxiliary tracing is started. It can have either of these values:

DFHAUXT|DFHBUXT

AUTOSWITCH_STATUS
Indicates whether or not an automatic switch to the inactive CICS auxiliary extent is to occur once only when the current extent fills up, or that such automatic switching should occur “continuously” whenever the current extent fills up. It can have any one of these values:

OFF|ONCE|CONTINUOUS

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

TRSR gate, START_GTF_TRACE function

The START_GTF_TRACE function of the TRSR gate is used to start the tracing of CICS activity to GTF. It is the responsibility of the user to ensure that GTF has been started in MVS with at least TRACE=USR. If it has not, CICS issues the GTF calls but they are ignored by GTF.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:

`CANT_GET_GTF_BUFFER`

meaning that there was insufficient storage for a buffer to be used in constructing continuation records when an individual entry is longer than 256 bytes.

TRSR gate, STOP_GTF_TRACE function

The STOP_GTF_TRACE function of the TRSR gate is used to stop tracing of CICS activity to GTF.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

TRSR gate, INQUIRE_GTF_TRACE function

The INQUIRE_GTF_TRACE function of the TRSR gate is used to return the current state of the GTF trace.

Input parameters

None.

Output parameters

GTF_STATUS
indicates whether CICS tracing to GTF is active (STARTED) or inactive (STOPPED).

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

TRSR gate, ACTIVATE_TRAP function

The ACTIVATE_TRAP function of the TRSR gate is used to activate the FE global trap/trace exit (DFHTRAP).

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DFHTRAP_NOT_FOUND DFHTRAP_UNUSABLE

RESPONSE	Possible REASON values
Note: DFHTRAP_NOT_FOUND indicates that the request to the CICS loader to acquire the FE global trap/trace exit program, DFHTRAP, has failed. DFHTRAP_UNUSABLE indicates that the trap was already active, but marked as unusable because a program check had previously occurred when DFHTRAP was in control.	

TRSR gate, DEACTIVATE_TRAP function

The DEACTIVATE_TRAP function of the TRSR gate is used to deactivate the FE global trap/trace exit (DFHTRAP).

Input parameters

None.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

Trace domain's generic gates

Table 109 summarizes the trace domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 109. Trace domain's generic gates

Gate	Trace	Function	Format
DMDM	TR 0001 TR 0002	PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
KETI	TR 0201 TR 0202	NOTIFY_RESET	KETI

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669

Format KETI—"Kernel domain's generic formats" on page 848

In preinitialization processing, the trace domain establishes the initial tracing status:

- A suitably sized internal trace table is created.
- If internal tracing or GTF tracing is required, set on the trace master flag.
- If required, start internal tracing and CICS GTF tracing.
- As required, set the auxiliary tracing switch status to 'started' or 'stopped'.

The information always comes from the system initialization parameters—trace domain is always cold started.

In initialization processing, the trace domain starts auxiliary tracing if it is required.

The trace domain does no quiesce processing.

In termination processing, the trace domain stops auxiliary tracing if it is active.

Control blocks

Figure 112 shows the control blocks associated with the trace domain.

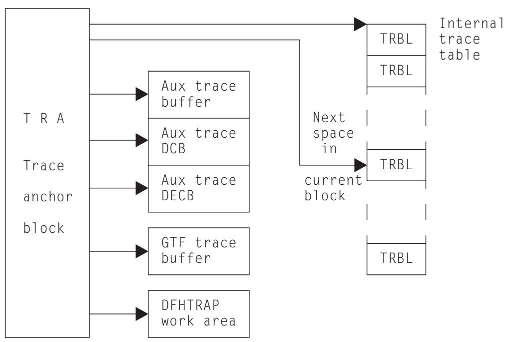


Figure 112. Control blocks associated with the trace domain

TR domain anchor block (TRA).
There is one TRA in the system. It contains all status information relating to the trace domain and also pointers to the other trace domain control blocks.

Internal trace table.
An area of virtual storage above the 16MB line used for recording trace entries.

TR block (TRBL).
The internal trace table consists of a number of TRBLs chained in a loop. They are each 4096 bytes long. Each block contains a standard header and a sequence of variable-length trace entries.

Trace entry (TREN).
All trace entries consist of a header together with any data specified on the call. The length of each trace entry is in the range 32 through 4072 bytes.

TR auxiliary trace data set DCB, DECB, and buffer.
During the auxiliary trace start process, an MVS GETMAIN is issued to acquire storage below the 16MB line for these areas. Their addresses are kept in the TRA. The storage is released when auxiliary trace is stopped.

GTF buffer.
During the GTF trace start process, an MVS GETMAIN is issued to acquire storage above the 16MB line for this area. It is 256 bytes long, and its address is kept in the TRA. The storage is released when GTF trace is stopped. The buffer is used when splitting large entries (more than 256 bytes) into 256-byte pieces to be written to GTF. This is done because GTF has a length restriction of 256 bytes.

Global trap/trace exit work area (TRGTW).
When the FE global trap/trace exit (DFHTRAP) is activated, an MVS GETMAIN is issued to acquire storage above the 16MB line for the TRGTW. This area contains a register save area and all working storage associated with DFHTRAP, including the parameter list passed to the exit program. Its address is kept in the TRA. The storage is released when the trap is deactivated.

See the *CICS Data Areas* manual for a detailed description of these control blocks.

Modules

Module	Function
DFHTRDM	Processes requests to the DMDM gate of the trace domain. Part of the DFHSIP load module.
DFHTRPT	Processes requests to the TRPT gate of the trace domain. Part of the DFHSIP load module.
DFHTRPX	Processes, within the calling domain, all TRACE_PUT requests that do not require special handling. Part of the DFHSIP load module.
DFHTRSR	Processes requests to the TRSR and KETI gates of the trace domain. Part of the DFHSIP load module.
DFHTRSU	Processes domain subroutine requests of format TRSU. Part of the DFHSIP load module.
DFHTRAO	Auxiliary trace output subroutines for interfacing with BSAM. Loaded separately below the 16MB line when auxiliary trace is started.
DFHTRAP	FE global trap/trace exit program. Loaded separately above the 16MB line when the trap is activated.

Copy books

Copy book	Function
DFHTRADS	Contains the definition of the parameter list passed to DFHTRAP.
DFHTRDS	Contains the definitions of the TRA and TRBL.
DFHTREN	Contains the definition of the trace entry (TREN) format.

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the trace domain are of the form TR xxxx; the corresponding trace levels are TR 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Dumps

A formatted system dump contains (depending on the options specified on the TR keyword):
- TR anchor block with interpretation
- Auxiliary trace data set DCB if data set open
- Auxiliary trace data set DECB if data set open
- Auxiliary trace buffer if data set open
- Internal trace table in abbreviated format
- Internal trace table in full format.

System dumps requested by the trace domain fall into two categories:

Dump code TRnnnn
These dump codes are preceded by a console message, DFHTRnnnn. See the *CICS Messages and Codes* manual for details.

Dump code KERNDUMP
At many points in its processing, the trace domain cannot issue domain calls because they would lead to further trace calls and possible recursion of the error. In these circumstances, the trace domain uses MVS WTO to

write a console message and the kernel dump function to take a system dump. All such dumps have dump code KERNDUMP. The message numbers for which this occurs are DFHTR0105, DFHTR0114, DFHTR0115, and DFHTR0116. See the *CICS Messages and Codes* manual for more details.

Chapter 110. Temporary storage domain (TS)

The temporary storage domain manages temporary storage requests.

Temporary storage domain's specific gates

Table 110 summarizes the temporary storage domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 110. Temporary storage domain's specific gates

Gate	Trace	Function	XPI
TSQR	TS 0201	WRITE	NO
	TS 0202	REWRITE	NO
		READ_INTO	NO
		READ_SET	NO
		READ_NEXT_INTO	NO
		READ_NEXT_SET	NO
		DELETE	NO
TSPT	TS 0301	PUT	NO
	TS 0302	PUT_REPLACE	NO
		GET	NO
		GET_SET	NO
		GET_RELEASE	NO
		GET_RELEASE_SET	NO
		RELEASE	NO
TSSH	TS 0A01	INITIALIZE	NO
	TS 0A02	INQUIRE_POOL_TOKEN	NO
		WRITE	NO
		REWRITE	NO
		READ_INTO	NO
		READ_SET	NO
		READ_NEXT_INTO	NO
		READ_NEXT_SET	NO
		DELETE	NO
		INQUIRE_SYSID_TABLE_TOKEN	NO
		START_BROWSE	NO
		GET_NEXT	NO
		END_BROWSE	NO
		INQUIRE_QUEUE	NO
TSSR	TS 0601	SET_START_TYPE	NO
	TS 0602	SET_BUFFERS	NO
		SET_STRINGS	NO
TSBR	TS 0701	INQUIRE_QUEUE	NO
	TS 0702	START_BROWSE	NO
		GET_NEXT	NO
		END_BROWSE	NO
		CHECK_PREFIX	NO

TSQR gate, WRITE function

If the queue does not exist, this function creates a queue with the single item provided, and the queue's "read cursor" is set to zero.

If the queue already exists, the item provided is appended to the queue, and the read cursor left unchanged.

Input parameters

QUEUE_NAME is the name of the queue being created or appended to.

ITEM_DATA is the address and length of the item being written.

[BMS] indicates whether or not BMS owns this queue. It can have either of these values:

YES|NO

SUSPEND indicates whether or not the request will be suspended if there is insufficient auxiliary storage to satisfy the request. This option is ignored if the queue is in main storage.

STORAGE_TYPE indicates whether the queue is to be created in main or auxiliary storage. Note that this option is ignored if the queue already exists.

[CALLER] indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:

EXEC|MACRO

[FMH] indicates whether the data contains an FMH. It can have either of these values:

YES|NO

Output parameters

[TOTAL_ITEMS]
is the total number of items in the queue on completion of the operation.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	QUEUE_FULL INSUFFICIENT_STORAGE INVALID_LENGTH IO_ERROR INVALID_QUEUE_TYPE LOCKED INVALID_QUEUE_NAME QUEUE_DELETED QUEUE_REMOTE

TSQR gate, REWRITE function

This function updates the specified item in an existing queue. The read cursor is unchanged.

Input parameters

QUEUE_NAME is the name of the queue being updated.

ITEM_NUMBER is the number of the item to be updated.

ITEM_DATA is the address and length of the item being written.

SUSPEND indicates whether the request will be suspended if there is insufficient auxiliary storage to satisfy the request. This option is ignored if the queue is in main storage.

[CALLER] indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:

EXEC|MACRO

[FMH] indicates whether the data contains an FMH. It can have either of these values:

YES|NO

Output parameters

[TOTAL_ITEMS]
is the total number of items in the queue.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INSUFFICIENT_STORAGE INVALID_LENGTH IO_ERROR INVALID_QUEUE_TYPE QUEUE_NOT_FOUND ITEM_NOT_FOUND LOCKED INVALID_QUEUE_NAME QUEUE_DELETED QUEUE_REMOTE

TSQR gate, READ_INTO function

This function reads the specified queue item into a buffer provided by the caller. The read cursor for the queue is set to the item number provided. The caller provides the address (item_buffer_p) and buffer length (item_buffer_m). The actual length of the record is returned in item_buffer_n. If item_buffer_n is greater than item_buffer_m, the data is truncated (but an OK response is returned).

Input parameters

QUEUE_NAME
is the name of the queue being read.

ITEM_NUMBER
is the number of the item to be read.

ITEM_BUFFER
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

[CALLER]
indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:

EXEC|MACRO

Output parameters

[TOTAL_ITEMS]
returns the total number of items in the queue.

[FMH]
indicates whether the data contains an FMH. It can have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IO_ERROR INVALID_QUEUE_TYPE QUEUE_NOT_FOUND ITEM_NOT_FOUND INVALID_QUEUE_NAME

TSQR gate, READ_SET function

This function reads the specified queue item into a storage area obtained by TS. The read cursor for the queue is set to the input item number.

Input parameters

QUEUE_NAME
is the name of the queue being read.

ITEM_NUMBER
is the number of the item to be read.

[TCTTE_ADDRESS]
is the address of the TCTTE - required if SET_STORAGE_CLASS(TERMINAL) is specified.

[SET_STORAGE_CLASS]
specifies the class of storage into which the item will be read. This may be either TASK (the default) or TERMINAL. If TERMINAL is specified, the item is read into a TIOA. It can have either of these values:

TASK|TERMINAL

[CALLER]
indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:

EXEC|MACRO

Output parameters

ITEM_DATA
returns the address and length of the item data.

[TOTAL_ITEMS]
returns the total number of items in the queue.

[FMH]
indicates whether the data contains an FMH. It can have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IO_ERROR INVALID_QUEUE_TYPE QUEUE_NOT_FOUND ITEM_NOT_FOUND INVALID_QUEUE_NAME

TSQR gate, READ_NEXT_INTO function

This function increments the read cursor by one and reads that item number into the buffer provided by the caller. The caller provides the address (item_buffer_p) and buffer length (item_buffer_m). The actual length of the record is returned in item_buffer_n. If item_buffer_n is greater than item_buffer_m, the data will have been truncated.

Input parameters

QUEUE_NAME
is the name of the queue being read.

ITEM_BUFFER
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

[CALLER]
indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:

EXEC|MACRO

ITEM NUMBER
returns the number of the item just read.

Output parameters

[TOTAL_ITEMS]
returns the total number of items in the queue.

[FMH]
indicates whether the data contains an FMH. It can have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IO_ERROR INVALID_QUEUE_TYPE QUEUE_NOT_FOUND ITEM_NOT_FOUND INVALID_QUEUE_NAME

TSQR gate, READ_NEXT_SET function

This function increments the queue's read cursor by one and reads that item number into a storage area obtained by TS.

Input parameters

QUEUE_NAME is the name of the queue being read.

[TCTTE_ADDRESS]
is the address of the TCTTE - required if SET_STORAGE_CLASS(TERMINAL) is specified.

[SET_STORAGE_CLASS]
specifies the type of storage into which the item will be read. This may be either TASK (the default) or TERMINAL. If TERMINAL is specified, the item is read into a TIOA. It can have either of these values:

TASK|TERMINAL

[CALLER] indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:

EXEC|MACRO

Output parameters

ITEM_DATA returns the address and length of the item data.

[ITEM_NUMBER]
returns the number of the item just read.

[TOTAL_ITEMS]
returns the total number of items in the queue.

[FMH] indicates whether the data contains an FMH. It can have either of these values:

YES|NO

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IO_ERROR INVALID_QUEUE_TYPE QUEUE_NOT_FOUND ITEM_NOT_FOUND INVALID_QUEUE_NAME

TSQR gate, DELETE function

This function deletes the specified queue.

Input parameters

QUEUE_NAME is the name of the queue to be deleted. the request.

[CALLER] indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:

EXEC|MACRO

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_QUEUE_TYPE QUEUE_NOT_FOUND LOCKED INVALID_QUEUE_NAME QUEUE_DELETED

TSQR gate, ALLOCATE_SET_STORAGE function

This function allocates set storage of the requested length.

Input parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_QUEUE_TYPE QUEUE_NOT_FOUND LOCKED INVALID_QUEUE_NAME QUEUE_DELETED

TSPT gate, PUT function

If the queue does not already exist, this function creates a queue with the single item provided.

If the queue already exists, and is recoverable, a duplicate_name exception is returned. Otherwise, the item is appended to the queue.

Input parameters

QUEUE_NAME is the name of the queue being created or appended to.

ITEM_DATA is the address and length of the item being written.

[IC_DATA] is the address and length of an optional ICE.

[BMS] this option indicates whether or not BMS owns this queue. If the queue already exists and is a BMS queue then BMS(YES) must be specified on the request. Otherwise an INVALID response is returned. It can have either of these values:

YES|NO

[IC] this option indicates whether or not Interval Control owns this queue. If the queue already exists and is an IC queue then IC(YES) must be specified on the request. Otherwise an INVALID response is returned. It can have either of these values:

YES|NO

[FMH] indicates whether the data contains an FMH. It can have either of these values:

YES|NO

SUSPEND indicates whether the request is to be suspended if there is insufficient auxiliary storage to satisfy the request.

Output parameters

RECOVERABLE returns whether the queue is recoverable or not.

QUEUE_CREATION_TIME
returns the store clock time at which the queue was created.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INSUFFICIENT_STORAGE QUEUE_FULL DUPLICATE_NAME INVALID_LENGTH IO_ERROR INVALID_QUEUE_TYPE LOCKED INVALID_QUEUE_NAME QUEUE_DELETED QUEUE_REMOTE

TSPT gate, PUT_REPLACE function

If the queue does not exist, this function creates the queue with the item provided. If the queue does exist, the first item in the queue is replaced by the item provided.

Input parameters

QUEUE_NAME is the name of the queue being created or written to.

ITEM_DATA is the address and length of the data item being written.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_LENGTH IO_ERROR INVALID_QUEUE_TYPE LOCKED INVALID_QUEUE_NAME QUEUE_DELETED QUEUE_REMOTE

TSPT gate, GET function

This function retrieves the first item in a "put" queue.

Input parameters

QUEUE_NAME is the name of the queue being accessed.

ITEM_BUFFER specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

Output parameters

[FMH] indicates whether the data contains an FMH. It can have either of these values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IO_ERROR INVALID_QUEUE_TYPE QUEUE_NOT_FOUND INVALID_QUEUE_NAME

TSPT gate, GET_SET function

This function retrieves the first item in a "put" queue into a set storage area.

Input parameters

QUEUE_NAME
is the name of the queue being accessed.

Output parameters

ITEM_DATA
returns the address and length of the item in set storage.

[FMH]
indicates whether the data contains an FMH. It can have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IO_ERROR INVALID_QUEUE_TYPE QUEUE_NOT_FOUND INVALID_QUEUE_NAME

TSPT gate, GET_RELEASE function

This function retrieves and deletes the first item in a "put" queue. If the queue has one item, the queue is deleted.

Input parameters

QUEUE_NAME
is the name of the queue being accessed.

ITEM_BUFFER
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

Output parameters

[FMH]
indicates whether the data contains an FMH. It can have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IO_ERROR INVALID_QUEUE_TYPE QUEUE_NOT_FOUND LOCKED INVALID_QUEUE_NAME QUEUE_DELETED

TSPT gate, GET_RELEASE_SET function

This function retrieves the first item in a "put" queue into set storage and then deletes it. If the queue has one item, the queue is deleted.

Input parameters

QUEUE_NAME
is the name of the queue being accessed.

Output parameters

ITEM_DATA
returns the address and length of the item in set storage.

[FMH]
indicates whether the data contains an FMH. It can have either of these values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IO_ERROR INVALID_QUEUE_TYPE QUEUE_NOT_FOUND LOCKED INVALID_QUEUE_NAME QUEUE_DELETED

TSPT gate, RELEASE function

This function deletes a "put" queue.

Input parameters

QUEUE_NAME is the name of the queue being deleted. the request.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_QUEUE_TYPE QUEUE_NOT_FOUND LOCKED INVALID_QUEUE_NAME QUEUE_DELETED

TSSH gate, INITIALIZE function

Initialize the Shared TS interface.

Input parameters

TSSH gate, INQUIRE_POOL_TOKEN function

Return token for the pool corresponding to the sysid provided.

Input parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SYSID_NOT_FOUND

TSSH gate, WRITE function

If the queue does not exist, this function creates a queue with the single item provided, and the queue's "read cursor" is set to zero.

If the queue already exists, the item provided is appended to the queue, and the read cursor left unchanged.

Input parameters

[POOL_TOKEN] is a token for the shared TS pool.
QUEUE_NAME is the name of the queue being created or appended to.
ITEM_DATA is the address and length of the item being written.
SUSPEND indicates whether or not the request will be suspended if there is insufficient storage to satisfy the request.
FMH indicates whether the data contains an FMH.
[TRANSID] is the id of the transaction which issued this request.
[TRANSACTION_NUMBER]
is the 4-byte transaction number (in packed-decimal format).

Output parameters

TOTAL_ITEMS is the total number of items in the queue on completion of the operation.
RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_ERROR IO_ERROR QUEUE_FULL INSUFFICIENT_STORAGE INVALID_LENGTH INVALID_QUEUE_NAME MAXIMUM_QUEUES_REACHED

TSSH gate, REWRITE function

This function updates the specified item in an existing queue. The read cursor is unchanged.

Input parameters

[POOL_TOKEN] is a token for the shared TS pool.
QUEUE_NAME is the name of the queue being updated.
ITEM_NUMBER is the number of the item to be updated.
ITEM_DATA is the address and length of the item being written.
SUSPEND indicates whether the request will be suspended if there is insufficient storage to satisfy the request.
FMH indicates whether the data contains an FMH.
[TRANSACTION_NUMBER]
is the 4-byte transaction number (in packed-decimal format).

Output parameters

TOTAL_ITEMS is the total number of items in the queue.
RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_ERROR IO_ERROR INSUFFICIENT_STORAGE INVALID_LENGTH QUEUE_NOT_FOUND ITEM_NOT_FOUND INVALID_QUEUE_NAME

TSSH gate, READ_INTO function

This function reads the specified queue item into a buffer provided by the caller. The read cursor for the queue is set to the item number provided. The caller provides the address (item_buffer_p) and buffer length (item_buffer_m). The actual length of the record is returned in item_buffer_n. If item_buffer_n is greater than item_buffer_m, the data is truncated (but an OK response is returned).

Input parameters

[POOL_TOKEN] is a token for the shared TS pool.

QUEUE_NAME is the name of the queue being read.

ITEM_NUMBER is the number of the item to be read.

ITEM_BUFFER specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

[TRANSACTION_NUMBER]
is the 4-byte transaction number (in packed-decimal format).

Output parameters

TOTAL_ITEMS returns the total number of items in the queue.

FMH indicates whether the data contains an FMH.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_ERROR IO_ERROR QUEUE_NOT_FOUND ITEM_NOT_FOUND INVALID_QUEUE_NAME

TSSH gate, READ_SET function

This function reads the specified queue item into a storage area obtained by TS. The read cursor for the queue is set to the input item number.

Input parameters

[POOL_TOKEN] is a token for the shared TS pool.

QUEUE_NAME is the name of the queue being read.

ITEM_NUMBER is the number of the item to be read.

[TRANSACTION_NUMBER]
is the 4-byte transaction number (in packed-decimal format).

Output parameters

ITEM_DATA returns the address and length of the item data.

TOTAL_ITEMS returns the total number of items in the queue.

FMH indicates whether the data contains an FMH.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	IO_ERROR SERVER_ERROR QUEUE_NOT_FOUND ITEM_NOT_FOUND INVALID_QUEUE_NAME

TSSH gate, READ_NEXT_INTO function

This function increments the read cursor by one and reads that item number into the buffer provided by the caller. The caller provides the address (item_buffer_p) and buffer length (item_buffer_m). The actual length of the record is returned in item_buffer_n. If item_buffer_n is greater than item_buffer_m, the data will have been truncated.

Input parameters

[POOL_TOKEN] is a token for the shared TS pool.

QUEUE_NAME is the name of the queue being read.

ITEM_BUFFER specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

[TRANSACTION_NUMBER]
is the 4-byte transaction number (in packed-decimal format).

ITEM NUMBER returns the number of the item just read.

Output parameters

TOTAL_ITEMS returns the total number of items in the queue.

FMH indicates whether the data contains an FMH.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_ERROR IO_ERROR QUEUE_NOT_FOUND ITEM_NOT_FOUND INVALID_QUEUE_NAME

TSSH gate, READ_NEXT_SET function

This function increments the queue's read cursor by one and reads that item number into a storage area obtained by TS.

Input parameters

[POOL_TOKEN] is a token for the shared TS pool.

QUEUE_NAME is the name of the queue being read.

[TRANSACTION_NUMBER]
is the 4-byte transaction number (in packed-decimal format).

Output parameters

ITEM_DATA returns the address and length of the item data.

ITEM_NUMBER returns the number of the item just read.

TOTAL_ITEMS returns the total number of items in the queue.

FMH indicates whether the data contains an FMH.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_ERROR IO_ERROR QUEUE_NOT_FOUND ITEM_NOT_FOUND INVALID_QUEUE_NAME

TSSH gate, DELETE function

This function deletes the specified queue.

Input parameters

[POOL_TOKEN] is a token for the shared TS pool.

QUEUE_NAME is the name of the queue to be deleted. the request.

[TRANSACTION_NUMBER]
is the 4-byte transaction number (in packed-decimal format).

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_ERROR IO_ERROR QUEUE_NOT_FOUND INVALID_QUEUE_NAME

TSSH gate, INQUIRE_SYSID_TABLE_TOKEN function

Input parameters

[POOL_TOKEN] is a token for the shared TS pool.

QUEUE_NAME is the name of the queue to be deleted. the request.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_ERROR IO_ERROR QUEUE_NOT_FOUND INVALID_QUEUE_NAME

TSSB gate, START_BROWSE function

Input parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	QUEUE_NOT_FOUND BROWSE_END SERVER_ERROR IO_ERROR

TSSB gate, GET_NEXT function

Returns information about the next queue in the browse.

Input parameters

None

Output parameters

QUEUE_NAME
is the name of the queue.

[LAST_REFERENCED_TIME]
is the time at which the queue was last referenced.

[TOTAL_ITEMS]
is the total number of items in the queue.

[TOTAL_LENGTH]
is the sum of the lengths of all the items in the queue.

[MAXIMUM_ITEM_LENGTH]
is the length of the longest item in the queue.

[MINIMUM_ITEM_LENGTH]
is the length of the shortest item in the queue.

[TRANSID]
is the id of the transaction whcih created the queue.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END SERVER_ERROR IO_ERROR

TSSB gate, END_BROWSE function

Ends the browse.

Input parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END SERVER_ERROR IO_ERROR

TSSB gate, INQUIRE_QUEUE function

Input parameters

[POOL_TOKEN]
is the token for the pool being inquired upon.

QUEUE_NAME
is the name of the queue being inquired upon.

[KEY_COMPARISON]
specifies the constraints on the inquire. The default is KEY_COMPARISON(EQ). It can have any one of these values:

EQ|GT|GTEQ

[TRANSACTION_NUMBER]
is the 4-byte transaction number (in packed-decimal format).

Output parameters

[OUTPUT_QUEUE_NAME]
is the name of the queue whose information is returned. Note that this may differ from queue_name unless key_comparison(eq) is specified.

[LAST_REFERENCED_TIME]
is the time at which the queue was last referenced.

[TOTAL_ITEMS]
is the total number of items in the queue.

[TOTAL_LENGTH]
is the sum of the lengths of all the items in the queue.

[MAXIMUM_ITEM_LENGTH]
is the length of the longest item in the queue.

[MINIMUM_ITEM_LENGTH]
is the length of the shortest item in the queue.

[TRANSID] is the id of the transaction which created the queue.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	QUEUE_NOT_FOUND SERVER_ERROR IO_ERROR

TSSR gate, SET_START_TYPE function

Input parameters

START indicates the type of startup requested.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

TSSR gate, SET_BUFFERS function

Sets the number of TS buffers to be used.

Input parameters

BUFFERS the number of buffers required.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

TSSR gate, SET_STRINGS function

This function sets the number of strings to be used.

Input parameters

STRINGS the number of strings to be used.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

TSBR gate, INQUIRE_QUEUE function

Input parameters

QUEUE_NAME is the name of the queue being inquired upon.

Output parameters

[CREATION_TIME]
is the time at which the queue was created.

[LAST_REFERENCED_TIME]
is the time at which the queue was last referenced.

[TRANSID] is the id of the transaction which created the queue.

[TOTAL_ITEMS]
is the total number of items in the queue.

[TOTAL_LENGTH]
is the sum of the lengths of all the items in the queue.

[MAXIMUM_ITEM_LENGTH]
is the length of the longest item in the queue.

[MINIMUM_ITEM_LENGTH]
is the length of the shortest item in the queue.

[STORAGE_TYPE]
indicates whether the queue is held in main or auxiliary storage. It can have either of these values:

MAIN|AUXILIARY

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	QUEUE_NOT_FOUND

TSBR gate, START_BROWSE function

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	QUEUE_NOT_FOUND

TSBR gate, GET_NEXT function

Returns information about the next queue in the browse.

Input parameters

None

Output parameters

QUEUE_NAME
is the name of the queue.

[CREATION_TIME]
is the time at which the queue was created.

[LAST_REFERENCED_TIME]
is the time at which the queue was last referenced.

[TRANSID]
is the id of the transaction which created the queue.

[TOTAL_ITEMS]
is the total number of items in the queue.

[TOTAL_LENGTH]
is the sum of the lengths of all the items in the queue.

[MAXIMUM_ITEM_LENGTH]
is the length of the longest item in the queue.

[MINIMUM_ITEM_LENGTH]
is the length of the shortest item in the queue.

[STORAGE_TYPE]
indicates whether the queue is held in main or auxiliary storage. It can have either of these values:

MAIN|AUXILIARY

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END

TSBR gate, END_BROWSE function

Ends the browse.

Input parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END

TSBR gate, CHECK_PREFIX function

Checks whether there are any queues with the prefix provided.

Input parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE NOT_FOUND

TSIC gate, DELIVER_IC_RECOVERY_DATA function

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

TSIC gate, INQUIRE_QUEUE function

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

TSIC gate, SOLICIT_INQUIRES function

This call is made from TS to IC to initiate inquire_queue requests from IC to TS.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

Temporary storage domain's generic gates

Table 111 summarizes the storage manager domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 111. Temporary storage domain's generic gates

Gate	Trace	Function	Format
DMDM	TS 0101 TS 0102	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	TS 0501 TS 0502	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
APUE	TS 0601 TS 0602	SET_EXIT_STATUS	APUE
RMRO	TS 0401 TS 0402	PERFORM_PREPARE PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT START_BACKOUT END_BACKOUT	RMRO
RMDE	TS 0401 TS 0402	START_DELIVERY DELIVER_RECOVERY END_DELIVERY	RMDE
RMKP	TS 0401 TS 0402	TAKE_KEYPOINT	RMKP

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format APUE—Chapter 70, "Application domain (AP)," on page 513
Format DMDM—Chapter 78, "Domain manager domain (DM)," on page 663
Format RMRO—Chapter 99, "Recovery Manager Domain (RM)," on page 1061
Format RMDE—Chapter 99, "Recovery Manager Domain (RM)," on page 1061
Format RMKP—Chapter 99, "Recovery Manager Domain (RM)," on page 1061
Format STST—Chapter 107, "Statistics domain (ST)," on page 1195

Modules

Module	Function
DFHTSDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHTSQR	Handles the following requests: WRITE REWRITE READ_INTO READ_SET READ_NEXT_INTO READ_NEXT_SET DELETE

Module	Function
DFHTSPT	Handles the following requests: PUT PUT_REPLACE GET GET_SET GET_RELEASE GET_RELEASE_SET RELEASE
DFHTSSH	Handles the following requests: INITIALIZE INQUIRE_POOL_TOKEN INQUIRE_SYSID_TABLE_TOKEN WRITE REWRITE READ_INTO READ_NEXT_INTO READ_SET READ_NEXT_SET DELETE START_BROWSE GET_NEXT END_BROWSE INQUIRE_QUEUE
DFHTSSR	Handles the following requests: SET_START_TYPE SET_BUFFERS SET_STRINGS SET_EXIT_STATUS
DFHTSRM	Handles the following requests: PERFORM_PREPARE PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT START_BACKOUT END_BACKOUT START_DELIVERY DELIVER_RECOVERY END_DELIVERY TAKE_KEYPOINT
DFHTSBR	Handles the following requests: INQUIRE_QUEUE START_BROWSE GET_NEXT END_BROWSE CHECK_PREFIX
DFHTSST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATISTICS
DFHTSDUF DFHTSDUC DFHTSDUS	TS domain offline dump formatting routines
DFHTSITR	Interprets TS domain trace entries

Exits

The temporary storage domain has four global user exit points: XTSQRIN, XTSQROUT, XTSPTIN and XTSPTOUT. For further information about these, see the *CICS Customization Guide*.

Trace

The point IDs for the temporary storage domain are of the form TS xxxx; the corresponding trace levels are TS 1, TS 2 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 111. User domain

The user domain provides an optional facility for checking user authority to sign on to a terminal.

User domain's specific gates

Table 112 summarizes the user domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 112. User domain's specific gates

Gate	Trace	Function	XPI
USAD	US 0201 US 0202	ADD_USER_WITH_PASSWORD ADD_USER_WITHOUT_PASSWORD DELETE_USER INQUIRE_USER INQUIRE_DEFAULT_USER VALIDATE_USER	NO NO NO NO NO NO
USFL	US 0501 US 0502	FLATTEN_USER UNFLATTEN_USER TAKEOVER	NO NO NO
USIS	US 0201 US 0202	SET_USER_DOMAIN_PARMS	NO
USXM	US 0401 US 0402	ADD_TRANSACTION_USER DELETE_TRANSACTION_USER END_TRANSACTION FLATTEN_TRANSACTION_USER INIT_TRANSACTION_USER INQUIRE_TRANSACTION_USER TERM_TRANSACTION_USER UNFLATTEN_TRANSACTION_USER	NO NO NO NO NO

USAD gate, ADD_USER_WITH_PASSWORD function

The ADD_USER_WITH_PASSWORD function of the USAD gate is used to add a user to the CICS region and verify the associated password or oidcard.

Input parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the length of the USERID value.

[PASSWORD_TYPE]
specifies if the password is masked. It can have either of these values:

`CLEAR|MASKED`

[PASSWORD]
is the current password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

[PASSWORD_LENGTH]
is the 8-bit length of the PASSWORD value. This parameter is only valid if PASSWORD is also specified.

[NEW_PASSWORD]
is a new password, 1 through 10 alphanumeric characters, to be assigned to the userid (specified by the USERID value). This parameter is only valid if PASSWORD is also specified.

[NEW_PASSWORD_LENGTH]
is the 8-bit length of the NEW_PASSWORD value. This parameter is only valid if NEW_PASSWORD is also specified.

[OIDCARD]
is an optional oidcard (operator identification card); a 65-byte field containing further security data from a magnetic strip reader (MSR) on 32xx devices.

[GROUPID]
is an optional identifier, 1 through 10 alphanumeric characters, of a RACF user group to which the userid (specified by the USERID value) is to be assigned.

[GROUPID_LENGTH]
is the 8-bit length of the GROUPID value. This parameter is only valid if GROUPID is also specified.

[ENTRY_PORT_NAME]
is an optional name of an entry port, 1 through 8 alphanumeric characters, to be assigned to the userid (specified by the USERID value).

[ENTRY_PORT_TYPE]
is the type of the optional entry port to be assigned to the userid (specified by the USERID value). It can have either of these values:

```
TERMINAL|CONSOLE
```

This parameter is only valid if ENTRY_PORT_NAME is also specified.

[SCOPE_CHECK]
indicates whether or not scope checking is to be performed for this function call. It can have either of these values:

```
YES|NO
```

SIGNON_TYPE
is the type of signon for the userid (specified by the USERID value). It can have any of these values:

```
ATTACH_SIGN_ON|DEFAULT_SIGN_ON|IRC_SIGN_ON|
LU61_SIGN_ON|LU62_SIGN_ON|NON_TERMINAL_SIGN_ON|
PRESET_SIGN_ON|USER_SIGN_ON|XRF_SIGN_ON
```

APPLID
is the application identifier for the CICS region.

Output parameters

USER_TOKEN
is the token identifying the userid in the user domain.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DEL_TIMEOUT_ENTRY_FAILED EXTRACT_FAILED GETMAIN_FAILED
EXCEPTION	ALREADY_SIGNED_ON PASSWORD_REQUIRED NEW_PASSWORD_REQUIRED OIDCARD_REQUIRED INVALID_USERID INVALID_PASSWORD INVALID_NEW_PASSWORD INVALID_OIDCARD INVALID_GROUPID INQUIRE_PW_DATA_FAILED USERID_NOT_IN_GROUP UNKNOWN_ESM_RESPONSE SECURITY_INACTIVE ESM_INACTIVE ENTRY_PORT_NOTAUTH APPLICATION_NOTAUTH USERID_REVOKED GROUP_ACCESS_REVOKED SECLABEL_CHECK_FAILED ESM_TRANQUIL ENQ_LIMIT_EXCEEDED

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION INVALID_PARAMETERS

USAD gate, ADD_USER_WITHOUT_PASSWORD function

The ADD_USER_WITHOUT_PASSWORD function of the USAD gate is used to add a user to the CICS region *without* verifying any password or oidcard.

Input parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the 8-bit length of the USERID value.

[APPLID]
is the application identifier for the CICS region.

[ENTRY_PORT_NAME]
is an optional name of an entry port, 1 through 8 alphanumeric characters, to be assigned to the userid (specified by the USERID value).

[ENTRY_PORT_TYPE]
is the type of the optional entry port to be assigned to the userid (specified by the USERID value). It can have either of these values:

```
TERMINAL|CONSOLE
```

This parameter is only valid if ENTRY_PORT_NAME is also specified.

[GROUPID]
is an optional identifier, 1 through 10 alphanumeric characters, of a RACF user group to which the userid (specified by the USERID value) is to be assigned.

[GROUPID_LENGTH]
is the 8-bit length of the GROUPID value. This parameter is only valid if GROUPID is also specified.

[SCOPE_CHECK]
indicates whether or not scope checking is to be performed for this function call. It can have either of these values:

```
YES|NO
```

SIGNON_TYPE
is the type of signon for the userid (specified by the USERID value). It can have any of these values:

```
ATTACH_SIGN_ON|DEFAULT_SIGN_ON|IRC_SIGN_ON|
LU61_SIGN_ON|LU62_SIGN_ON|NON_TERMINAL_SIGN_ON|
PRESET_SIGN_ON|USER_SIGN_ON|XRF_SIGN_ON
```

[SUSPEND]
indicates whether a wait during add user processing is acceptable. It can have either of these values:

```
YES|NO
```

[UUID]
is the unique universal ID (UUID) for the user.

Output parameters

USER_TOKEN
is the token identifying the userid in the user domain.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DEL_TIMEOUT_ENTRY_FAILED EXTRACT_FAILED GETMAIN_FAILED

RESPONSE	Possible REASON values
EXCEPTION	ALREADY_SIGNED_ON APPLICATION_NOTAUTH ENTRY_PORT_NOTAUTH ESM_INACTIVE ESM_TRANQUIL GROUP_ACCESS_REVOKED INVALID_GROUPID INVALID_USERID SECLABEL_CHECK_FAILED SECURITY_INACTIVE UNKNOWN_ESM_RESPONSE USER_NOT_LOCATED USERID_NOT_IN_GROUP USERID_REVOKED ENQ_LIMIT_EXCEEDED
INVALID	INVALID_FORMAT INVALID_FUNCTION INVALID_PARAMETERS

USAD gate, DELETE_USER function

The DELETE_USER function of the USAD gate is used to delete the user from the CICS region.

Input parameters

USER_TOKEN
is the token identifying the userid in the user domain.

SIGNOFF_TYPE
is the type of signoff for the userid identified by the SECURITY_TOKEN value. It can have any of these values:

```
ABNORMAL_SIGN_OFF|ATTACH_SIGN_OFF|DEFERRED_SIGN_OFF|
DELETE_SIGN_OFF|LINK_SIGN_OFF|NON_TERMINAL_SIGN_OFF|
PRESET_SIGN_OFF|UNFLATTEN_USER_SIGN_OFF|
USER_SIGN_OFF|XRF_SIGN_OFF
```

DELETE_IMMEDIATE
indicates whether the user should be deleted immediately. It can have one of these values:

```
YES|NO
```

Output parameters

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ADD_TIMEOUT_ENTRY_FAILED FREEMAIN_FAILED
EXCEPTION	INVALID_USER_TOKEN DEFAULT_USER_TOKEN SECURITY_INACTIVE ESM_TRANQUIL ESM_INACTIVE UNKNOWN_ESM_RESPONSE

USAD gate, INQUIRE_USER function

The INQUIRE_USER function of the USAD gate is used to inquire about the attributes of the user represented by the user token.

Input parameters

USER_TOKEN
is the token identifying the userid to the user domain.

Output parameters

[USERID]
is the identifier of the user (a userid of 1 through 10 alphanumeric characters).

[USERID_LENGTH]
is the length of the USERID value.

[USERNAME]
is an optional buffer into which the attributes of the user are placed.

[CURRENT_GROUPID]
is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid (specified by the SECURITY_TOKEN value) is assigned.

[CURRENT_GROUPID_LENGTH]
is the 8-bit length of the GROUPID value.

[NATIONAL_LANGUAGE]
is a three-character code identifying the national language for the userid. It can have any of the values in Table 120 on page 1318.

[OPERATOR_CLASSES]
identifies the operator classes to which the user belongs. This is a 24-bit value, with each bit determining whether or not the user is a member of that class.

[OPERATOR_IDENT]
is the operator identification code, 1 through 3 alphanumeric characters, for the userid.

[ENTRY_PORT_NAME]
is the name of the entry port assigned to the userid.

[ENTRY_PORT_TYPE]
is the type of the entry port assigned to the userid. It can have either of these values:

`TERMINAL|CONSOLE`

This parameter is only valid if ENTRY_PORT_NAME is also specified.

[OPERATOR_PRIORITY]
is the operator priority value, in the range 0 through 255 (where 255 is the highest priority), for the userid.

[TIMEOUT]
is the number of minutes, in the range 0 through 60, that must elapse since the user last used the terminal before CICS "times-out" the terminal.

Notes:
1. CICS rounds values up to the nearest multiple of 5.
2. A TIMEOUT value of 0 means that the terminal is not timed out.

[XRF_REFLECTABLE]
indicates whether or not you want CICS to sign off the userid following an XRF takeover. It can have either of these values:

`YES|NO`

[ACEE_PTR]
is a pointer to the access control environment element, the control block that is generated by an external user (ESM) when the user signs on. If the user is not signed on, the address of the CICS DFLTUSER's ACEEis returned. If an ACEE does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_USER_TOKEN

USAD gate, INQUIRE_DEFAULT_USER function

The INQUIRE_DEFAULT_USER function of the USAD gate is used to inquire about the attributes of the default user (specified on the DFLTUSER system initialization parameter).

Input parameters

None

Output parameters

[USERID]
is the identifier of the user (a userid of 1 through 10 alphanumeric characters).

[USERID_LENGTH]
is the length of the USERID value.

[USERNAME]
is an optional buffer into which the attributes of the default user are placed.

[CURRENT_GROUPID]
is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid (specified by the SECURITY_TOKEN value) is assigned.

[CURRENT_GROUPID_LENGTH]
is the 8-bit length of the GROUPID value.

[NATIONAL_LANGUAGE]
is a three-character code identifying the national language for the userid. It can have any of the values in Table 120 on page 1318.

[OPERATOR_CLASSES]
identifies the operator classes to which the user belongs. This is a 24-bit value, with each bit determining whether or not the user is a member of that class.

[OPERATOR_IDENT]
is the operator identification code, 1 through 3 alphanumeric characters, for the userid.

[OPERATOR_PRIORITY]
is the operator priority value, in the range 0 through 255 (where 255 is the highest priority), for the userid.

[TIMEOUT]
is the number of minutes, in the range 0 through 60, that must elapse since the user last used the terminal before CICS "times-out" the terminal.

Notes:
1. CICS rounds values up to the nearest multiple of 5.
2. A TIMEOUT value of 0 means that the terminal is not timed out.

[XRF_REFLECTABLE]
indicates whether or not you want CICS to sign off the userid following an XRF takeover. It can have either of these values:

`YES|NO`

[ACEE_PTR]
is a pointer to the access control environment element, the control block that is generated by an external user (ESM) when the default user signs on. If an ACEE does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

USAD gate, VALIDATE_USERID function

The VALIDATE_USERID function of the USAD gate is used to verify that the specified userid is a valid userid.

Input parameters

[USERID]
is the userid to be validated.

[USERID_LENGTH]
is the length of the userid to be validated.

Output parameters

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SECURITY_INACTIVE USERID_NOT_DEFINED USERID_NOT_DETERMINED

USFL gate, FLATTEN_USER function

The FLATTEN_USER function of the USFL gate is used to flatten the user's security state and place into the FLATTENED_USER buffer provided.

Input parameters

USER_TOKEN
is the token identifying the userid.

FLATTENED_USER
is the buffer into which the flattened security state is placed.

Output parameters

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP DIR_MANAGER_LOCATE_FAILED SEC_DOM_FLATTEN_FAILED
EXCEPTION	INVALID_USER_TOKEN SECURITY_INACTIVE ESM_INACTIVE ESM_TRANQUIL UNKNOWN_ESM_RESPONSE
INVALID	INVALID_FORMAT INVALID_FUNCTION INVALID_FLATTENED_BUFFER

USFL gate, TAKEOVER function

The TAKEOVER function of the USFL gate is used, when an XRF takeover occurs, to obtain the SNSCOPE ENQ resources for those users who could not obtain it during tracking, because the resources were already held by the active region.

Input parameters

None.

Output parameters

RESPONSE
is the domains response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

USFL gate, UNFLATTEN_USER function

The UNFLATTEN_USER function of the USFL gate is used to unflatten the user security state data in the FLATTENED_USER buffer, and add the userid to the user domain.

Input parameters

FLATTENED_USER
is a buffer containing flattened security state data for a userid.

Output parameters

USER_TOKEN
is the token identifying the userid in the user domain.

RESPONSE
is the domains response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP DEL_TIMEOUT_ENTRY_FAILED DIR_MANAGER_ADD_FAILED DIR_MANAGER_DELETE_FAILED FREEMAIN_FAILED GETMAIN_FAILED SEC_DOMAIN_DELETE_FAILED SEC_DOM_UNFLATTEN_FAILED
EXCEPTION	ALREADY_SIGNED_ON APPLICATION_NOTAUTH ENTRY_PORT_NOTAUTH ESM_INACTIVE ESM_TRANQUIL GROUP_ACCESS_REVOKED SECLABEL_CHECK_FAILED SECURITY_INACTIVE UNKNOWN_ESM_RESPONSE USERID_NOT_IN_GROUP USERID_REVOKED USERID_UNDEFINED
INVALID	INVALID_FLATTENED_BUFFER INVALID_FORMAT INVALID_FUNCTION

USIS gate, SET_USER_DOMAIN_PARMS function

At CICS startup, loads information for the user domain from the system initialization table (SIT) into the user state data.

Input parameters

DEFAULT_USERID
is the default userid, as 1 through 10 alphanumeric characters.

SIGNON_SCOPE
is the scope for which the default userid can be signed on. It can have any of these values:

NONE|CICS|MVSIMAGE|SYSPLEX

DIRECTORY_TIMEOUT_VALUE
is the intersystem refresh delay, in the range 0 through 10080 minutes (up to 7 days), for the default userid.

APPLID
is the application identifier for the CICS region.

Output parameters

RESPONSE
is the domains response to the call. It can have any of these values:

OK|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

USXM gate, ADD_TRANSACTION_USER function

The ADD_TRANSACTION_USER function of the USXM gate sets the user characteristics (as security tokens) for a transaction.

Input parameters

[PRINCIPAL_USER_TOKEN]
is the optional principal user token representing the characteristics of the principal user of the transaction.

[SESSION_USER_TOKEN]
is the optional session user token representing the characteristics of the session user of the transaction.

[EDF_USER_TOKEN]
is the optional EDF user token representing the characteristics of the EDF user of the transaction.

Output parameters

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	ALREADY_SIGNED_ON DUPLICATE_USER INVALID_USER_TOKEN
INVALID	INVALID_FORMAT INVALID_FUNCTION

USXM gate, DELETE_TRANSACTION_USER function

The DELETE_TRANSACTION_USER function of the USXM gate deletes the user token of the specified token type for the transaction.

Input parameters

TOKEN_TYPE
is the type of user token for the transaction. It can have any of these values:

PRINCIPAL|SESSION|EDF

Output parameters

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	NO_USER_TOKEN
INVALID	INVALID_FORMAT INVALID_FUNCTION

USXM gate, END_TRANSACTION function

The END_TRANSACTION function of the USXM gate deletes all the user token to security token maps for the transaction.

Input parameters

None.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND FREEMAIN_FAILED LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

USXM gate, FLATTEN_TRANSACTION_USER function

The FLATTEN_TRANSACTION_USER function of the USXM gate creates the contents of a FLAT_TRANSUSER buffer from the principal user of the current transaction.

Input parameters

FLAT_TRANSUSER
is the buffer to be created.

Output parameters

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FLAT_TRANSUSER

USXM gate, INIT_TRANSACTION_USER function

The INIT_TRANSACTION_USER function of the USXM gate initializes the transaction for the user characteristics identified by the PRINCIPAL_USER_TOKEN value.

Input parameters

PRINCIPAL_USER_TOKEN is the principal user token representing the characteristics of the principal user of the transaction.

[SESSION_USER_TOKEN] is the optional session user token representing the characteristics of the session user of the transaction.

[EDF_USER_TOKEN] is the optional EDF user token representing the characteristics of the EDF user of the transaction.

[XMAT_CALL] indicates whether the function is called while a transaction is being attached. It can have either of these values:

YES|NO

Output parameters

USDOM_TRANSACTION_TOKEN
is the user token to be used for reference to user characteristics only. It is treated as the principal user token until the next ADD_TRANSACTION_USER call for the transaction.

PRIORITY is the priority value, in the range 0 through 255 (where 255 is the highest priority), for the user with the token identified by the PRINCIPAL_USER_TOKEN value.

RESPONSE is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND GETMAIN_FAILED LOOP
EXCEPTION	INVALID_USER_TOKEN
INVALID	INVALID_FORMAT INVALID_FUNCTION

USXM gate, INQUIRE_TRANSACTION_USER function

The INQUIRE_TRANSACTION_USER function of the USXM gate inquires about the user characteristics associated with the transaction identified by the USDOM_TRANSACTION_TOKEN value.

Input parameters

USDOM_TRANSACTION_TOKEN
is the user token to be used for reference to user characteristics only.

Output parameters

[USERID]
is the identifier of the user (a userid of 1 through 10 alphanumeric characters).

USERID_LENGTH
is the length of the USERID value.

[USERNAME]
is an optional buffer that contains the attributes of the user.

[CURRENT_GROUPID]
is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the user is assigned.

[CURRENT_GROUPID_LENGTH]
is the 8-bit length of the GROUPID value.

[NATIONAL_LANGUAGE]
is a three-character code identifying the national language for the user. It can have any of the values in Table 120 on page 1318.

[OPERATOR_CLASSES]
identifies the operator classes to which the user belongs. This is a 24-bit value, with each bit determining whether or not the user is a member of that class.

[OPERATOR_IDENT]
is the operator identification code, 1 through 3 alphanumeric characters, for the user.

[ENTRY_PORT_NAME]
is the name of the entry port assigned to the userid.

[ENTRY_PORT_TYPE]
is the type of the entry port assigned to the userid. It can have either of these values:

```
TERMINAL|CONSOLE
```

This parameter is only valid if ENTRY_PORT_NAME is also specified.

[APPLID]
is the application identifier for the CICS region.

[OPERATOR_PRIORITY]
is the operator priority value, in the range 0 through 255 (where 255 is the highest priority), for the user.

[TIMEOUT]
is the number of minutes, in the range 0 through 60, that must elapse since the user last used the terminal before CICS "times-out" the terminal.

Notes:
1. CICS rounds values up to the nearest multiple of 5.
2. A TIMEOUT value of 0 means that the terminal is not timed out.

[XRFSOFF]
indicates whether or not you want CICS to sign off the user following an XRF takeover. It can have either of these values:

```
YES|NO
```

[ACEE_PTR]
is a pointer to the access control environment element, the control block that is generated by an external user (ESM) when the user signs on. If the user is not signed on, the address of the CICS DFLTUSER's ACEEis returned. If an ACEE does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

PRINCIPAL_USER_TOKEN
is the token identifying the userid in the user domain.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR
```

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

USXM gate, TERM_TRANSACTION_USER function

The TERM_TRANSACTION_USER function of the USXM gate removes the state information created by an INIT_TRANSACTION_USER function.

Input parameters

USDOM_TRANSACTION_TOKEN
is the token that identifies the state data to be removed.

Output parameters

RESPONSE
is the domains response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND FREEMAIN_FAILED LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

USXM gate, UNFLATTEN_TRANSACTION_USER function

The UNFLATTEN_TRANSACTION_USER function of the USXM gate adds (by the ADD_USER_WITHOUT_PASSWORD function of the USAD gate) the user defined by the contents of the supplied FLAT_TRANSUSER buffer.

Input parameters

FLAT_TRANUSER
is the buffer containing data that defines the user to be added.

[SUSPEND]
indicates whether a wait during add user processing is acceptable. It can have either of these values:

`YES|NO`

Output parameters

PRINCIPAL_USER_TOKEN
is the token identifying the userid in the user domain.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

RESPONSE	Possible REASON values
EXCEPTION	APPLICATION_NOTAUTH ENTRY_PORT_NOTAUTH ESM_INACTIVE ESM_TRANQUIL GROUP_ACCESS_REVOKED INVALID_GROUPID INVALID_USERID SECLABEL_CHECK_FAILED SECURITY_INACTIVE UNKNOWN_ESM_RESPONSE USER_NOT_LOCATED USERID_NOT_IN_GROUP USERID_REVOKED

User domain's generic gates

Table 113 summarizes the user domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 113. User domain's generic gates

Gate	Trace	Function	Format
DMDM	US 0101 US 0102	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	US 0601 US 0602	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669

Format STST—"Statistics domain's generic format" on page 1198

In initialization processing, performs internal routines to set up the user domain, and gets the initial user options, as for "USIS gate, SET_USER_DOMAIN_PARMS function" on page 1249.

For a cold start, the user options come from the system initialization parameters; for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

User domain also issues console messages during initialization to report whether or not security is active.

In quiesce and termination processing, the user domain performs only internal routines.

Modules

Module	Function
DFHUSAD	Handles the following requests: ADD_USER_WITH_PASSWORD ADD_USER_WITHOUT_PASSWORD DELETE_USER INQUIRE_USER INQUIRE_DEFAULT_USER VALIDATE_USERID
DFHUSDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHUSDUF	US domain offline dump formatting routine
DFHUSFL	Handles the following requests: FLATTEN_USER UNFLATTEN_USER TAKEOVER
DFHUSIS	Handles the following requests: SET_USER_DOMAIN_PARMS
DFHUSST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHUSTI	Handles user timeout processing.
DFHUSTRI	Interprets US domain trace entries
DFHUSXM	Handles the following requests: ADD_TRANSACTION_USER DELETE_TRANSACTION_USER END_TRANSACTION INIT_TRANSACTION_USER INQUIRE_TRANSACTION_USER FLATTEN_TRANSACTION_USER UNFLATTEN_TRANSACTION_USER

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the user domain are of the form US xxxx; the corresponding trace levels are US 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 112. Web domain (WB)

The Web domain manages interaction betwen CICS and Web clients, or between CICS as an HTTP client and servers on the Internet. For more information, see *CICS Internet Guide*.

Web domain's specific gates

Table 114 summarizes the Web domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 114. Web domain's specific gates

Gate	Trace	Function	XPI
WBAP	WB 0300	START_BROWSE	NO
	WB 0301	READ_NEXT	NO
		END_BROWSE	NO
		GET_MESSAGE_BODY	NO
		GET_HTTP_RESPONSE	NO
		SEND_RESPONSE	NO
		READ_HEADER	NO
		WRITE_HEADER	NO
		INQUIRE	NO
WBCL	WB 0700	PARSE_URL	NO
	WB 0701	OPEN_SESSION	NO
		WRITE_HEADER	NO
		WRITE_REQUEST	NO
		READ_RESPONSE	NO
		READ_HEADER	NO
		START_BROWSE_HEADERS	NO
		READ_NEXT_HEADER	NO
		END_BROWSE_HEADERS	NO
		INQUIRE_SESSION	NO
		CLOSE_SESSION	NO
WBRP	WB 0A00	CATALOG_URIMAP	NO
	WB 0A01	DELETE_URIMAP	NO
		CATALOG_HOST	NO
		DELETE_HOST	NO
		RECOVER_DEFINITIONS	NO
WBSR	WB 0500	SEND	NO
	WB 0501	RECEIVE	NO
		SEND_STATIC_RESPONSE	NO
WBUR	WB 0900	INITIALIZE_URIMAPS	NO
	WB 0901	ADD_REPLACE_URIMAP	NO
		DELETE_URIMAP	NO
		LOCATE_URIMAP	NO
		INQUIRE_URIMAP	NO
		SET_URIMAP	NO
		START_BROWSE_URIMAP	NO
		GET_NEXT_URIMAP	NO
		END_BROWSE_URIMAP	NO
		INQUIRE_HOST	NO
		SET_HOST	NO
		START_BROWSE_HOST	NO
		GET_NEXT_HOST	NO
		END_BROWSE_HOST	NO

WBAP gate, START_BROWSE function

The START_BROWSE function starts a browse of the HTTP headers or the HTML forms data in an HTTP request.

Input parameters

None

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_REQUEST_FORMAT NON_WEB_TRANSACITON HEADER_BROWSE_ACTIVE FORMFLD_BROWSE_ACTIVE CLIENT_CODEPAGE_UNSUPP SERVER_CODEPAGE_UNSUPP NO_FORMS_DATA INVALID_CODEPAGE_COMBIN
DISASTER	FORMFIELD_STRUCT_FORM_ERR FORMFIELD_CANNOT_GET_BODY FORMFIELD_STRUCT_CORRUPT FORMFIELD_CORRUPT_HEADER FORMFIELD_NO_BOUNDARY_STR FORMFIELD_NO_CONTENT_HDR FORMFIELD_UNKNOWN_FORMTYPE NO_CONVERT_PARM

WBAP gate, READ_NEXT function

The READ_NEXT function returns the next HTTP header in a browse of HTTP headers.

Input parameters

None

Output parameters

HTTP_HEADER_BUFFER_NAME
returns the name of the next HTTP header

HTTP_HEADER_BUFFER_VALUE
returns the value of the next HTTP header

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END NON_WEB_TRANSACTION HEADER_BROWSE_NOT_ACTIVE FORMFLD_BROWSE_NOT_ACTIVE HEADER_VALUE_LENGTH_ERROR HEADER_NAME_LENGTH_ERROR INVALID_HEADER FORMFLD_VALUE_LENGTH_ERROR FORMFLD_NAME_LENGTH_ERROR NO_FORMS_DATA INVALID_FORMFLD
DISASTER	FORMFIELD_STRUCT_CORRUPT FORMFIELD_CORRUPT_HEADER NO_CONVERT_PARM

WBAP gate, END_BROWSE function

The END_BROWSE function defines the end of a browse of the HTTP headers received for an HTTP request.

Input parameters

None

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NON_WEB_TRANSACTION HEADER_BROWSE_NOT_ACTIVE FORMFLD_BROWSE_NOT_ACTIVE

WBAP gate, GET_MESSAGE_BODY function

The GET_MESSAGE_BODY function retrieves the previously constructed body of an HTTP response.

Input parameters

DATA_BUFFER
Buffer into which the data is to be placed

TRUNCATE
indicates whether or not data is to be truncated if the buffer is too small. It can have the following values:
YES|NO

CLIENT_CODEPAGE
ASCII Codepage into which the data is to be converted before being passed back to the caller

SERVER_CODEPAGE
EBCDIC Codepage of the data to be passed back

CONVERT
indicates whether or not data is to undergo codepage conversion. It can have the following values:
YES|NO

Output parameters

SET_BLOCK
Address of a block of storage containing the message body

REQUEST_TYPE
Indicates whether we are processing an HTTP Request It can have the following values:
HTTP|NON_HTTP

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CODEPAGE_NOT_FOUND NON_WEB_TRANSACTION

WBAP gate, GET_HTTP_RESPONSE function

The GET_HTTP_RESPONSE function retrieves the HTTP Response which has been constructed by a Web API application program.

Input parameters

DOCUMENT_TOKEN
is the 8 byte field into which CICS places the document token identifying the document which contains the body of the HTTP response

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NON_WEB_TRANSACTION NO_PREVIOUS_WEB_SEND

WBAP gate, SEND_RESPONSE function

The SEND_RESPONSE function identifies a CICS Document which is to be used as the body of a HTTP response, and the HTTP reason code with which that response is to be returned.

Input parameters

DOCUMENT_TOKEN
identifies the CICS document to be used as the body of the HTTP response

CLIENT_CODEPAGE
identifies the ASCII codepage into which the body of the HTTP response is to be converted

STATUS_CODE
HTTP response code with which the HTTP response is returned

STATUS_TEXT
Text to accompany HTTP response code with which the HTTP response is returned.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NON_WEB_TRANSACTION DOCUMENT_NOT_FOUND CODEPAGE_NOT_FOUND

WBAP gate, READ_HEADER function

The READ_HEADER function returns the value of a specific HTTP request header.

Input parameters

HTTP_HEADER_BLOCK_VALUE
Buffer containing name of the header for which a value is returned

HTTP_HEADER_BLOCK_VALUE
Buffer containing the value of the requested header

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	HEADER_NOT_FOUND INVALID_REQUEST_FORMAT NON_WEB_TRANSACTION

WBAP gate, WRITE_HEADER function

The WRITE_HEADER function causes a HTTP response header to be stored by CICS.

Input parameters

HTTP_HEADER_BLOCK_NAME
Buffer containing name of header

HTTP_HEADER_BLOCK_VALUE
Buffer containing value of header

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NON_WEB_TRANSACTION

WBAP gate, INQUIRE function

The INQUIRE function passes back information pertaining to an HTTP request.

Input parameters

SERVER_NAME
Buffer to contain TCP/IP name of CICS

CLIENT_NAME
Buffer to contain TCP/IP name of client from which HTTP request was received.

HTTP_METHOD
Buffer to contain HTTP method specified on the HTTP request

HTTP_VERSION
Buffer to contain HTTP version specified on the HTTP request

QUERYSTRING
Buffer to contain HTTP query string specified on the HTTP request

URI
Buffer to contain URI specified on the HTTP request

Output parameters

CLIENT_ADDR
Fullword containing IP address of the client from which the HTTP request was received

SERVER_ADDR
Fullword containing IP address of the TCP/IP stack on which the HTTP request was received

SERVER_PORT
Fullword containing port number on which the HTTP request was received

CERTIFICATE_TOKEN
eight byte token identifying SSL certificate of client issuing this HTTP request

REQUEST_TYPE
Indicates whether CICS recognized the incoming data as a valid HTTP request. Can be set to:

HTTP|NON_HTTP

SSL_TYPE
Indicates what level of SSL support applies to the incoming HTTP request. Can be set to:

YES|NO|CLIENTAUTH

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NON_WEB_TRANSACTION INVALID_REQUEST_FORMAT

WBCL gate, PARSE_URL function

The PARSE_URL function parses a URL into its constituent components.

Input parameters
URL

Output parameters
SCHEME
[SCHEME_NAME]
HOST
PORT
PATH
QUERY_STRING
[IP_ADDRESS]

RESPONSE is the domain's response to the call. It can have any of these values:
`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBCL gate, OPEN_SESSION function

The OPEN_SESSION function opens a session with the HTTP server.

Input parameters
SCHEME
HOST
PORT
[URIMAP]
[HOST_CODEPAGE]
[CERTIFICATE_LABEL]
[CIPHER_COUNT, CIPHER_SUITES]
[PROXY_URL]

Output parameters
SESSION_TOKEN
[HTTP_VNUM]
[HTTP_RNUM]

RESPONSE is the domain's response to the call. It can have any of these values:
`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBCL gate, WRITE_HEADER function

The WRITE_HEADER function adds one HTTP header to the HTTP request being composed. It can be called multiple times to add multiple headers.

Input parameters
SESSION_TOKEN
NAME
VALUE

Output parameters
RESPONSE is the domain's response to the call. It can have any of these values:
`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBCL gate, WRITE_REQUEST function

The WRITE_REQUEST function appends the request body to the HTTP request being composed, and schedules it to be sent. It also handles sending a chunk of data.

Input parameters

SESSION_TOKEN
[CHUNK | BODY | CONTAINER_NAME CONTAINER_POOL]
[DOCUMENT_TOKEN]
[PATH]
[URIMAP]
[QUERY_STRING]
METHOD
[MEDIATYPE]
[ACTION_PARAMETER]
[TRANSLATE]
[CHARSET]
[HOST_CODEPAGE]
[ACTION]
[CLOSE]
[CONVERSE]

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION.

WBCL gate, READ_RESPONSE function

The READ_RESPONSE function waits for and then reads the HTTP response that is expected from the HTTP server.

Input parameters

SESSION_TOKEN
[BODY | CONTAINER_NAME CONTAINER_POOL]
[TRANSLATE]
[HOST_CODEPAGE]
[TIME_OUT_VALUE]
[TRUNCATE]
[MAX_DATA_LENGTH]
[STATUS_TEXT]

Output parameters

STATUS_CODE
[SET_BUFFER]
[MEDIATYPE]
[CHARSET]
RESPONSE
is the domain's response to the call. It can have any of these values:
`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION.

WBCL gate, READ_HEADER function

The READ_HEADER function reads a specific HTTP header from the HTTP response that has been received.

Input parameters

SESSION_TOKEN
NAME
VALUE_BUFFER

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

	OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED
[REASON]	is returned when RESPONSE is EXCEPTION.

WBCL gate, START_BROWSE_HEADERS function

The START_BROWSE_HEADERS function starts a browse of the HTTP headers for a response that has been received.

Input parameters

SESSION_TOKEN

Output parameters

RESPONSE	is the domain's response to the call. It can have any of these values:
	OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED
[REASON]	is returned when RESPONSE is EXCEPTION.

WBCL gate, READ_NEXT_HEADER function

The READ_NEXT_HEADER function reads the next HTTP header in the browse operation for an HTTP response that has been received.

Input parameters

SESSION_TOKEN
NAME_BUFFER
VALUE_BUFFER

Output parameters

RESPONSE	is the domain's response to the call. It can have any of these values:
	OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED
[REASON]	is returned when RESPONSE is EXCEPTION.

WBCL gate, END_BROWSE_HEADERS function

The END_BROWSE_HEADERS function ends a browse of the HTTP headers for an HTTP response that has been received.

Input parameters

SESSION_TOKEN

Output parameters

RESPONSE	is the domain's response to the call. It can have any of these values:
	OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED
[REASON]	is returned when RESPONSE is EXCEPTION.

WBCL gate, INQUIRE_SESSION function

The INQUIRE_SESSION function returns information about the specified connection to a server, represented by the session token.

Input parameters

SESSION_TOKEN
[HOST_BUFFER]
[PATH_BUFFER]

Output parameters

[SCHEME]
[HTTP_VNUM]
[HTTP_RNUM]
[URIMAP]
[PORT]

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBCL gate, CLOSE_SESSION function

The CLOSE_SESSION function ends the connection to the server by closing the socket and releasing the session control block.

Input parameters

SESSION_TOKEN

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBRP gate, CATALOG_URIMAP function

The CATALOG_URIMAP function adds a URIMAP resource definition to the catalog.

Input parameters

URIMAP_DEFINITION

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LOOP ABEND CATALOG_WRITE_FAILURE

WBRP gate, DELETE_URIMAP function

The DELETE_URIMAP function deletes a URIMAP resource definition from the catalog.

Input parameters

URIMAP

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LOOP ABEND CATALOG_DELETE_FAILURE

WBRP gate, CATALOG_HOST function

The CATALOG_HOST function adds a virtual host to the catalog.

Input parameters

HOST_DEFINITION

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LOOP ABEND CATALOG_WRITE_FAILURE

WBRP gate, DELETE_HOST function

The DELETE_HOST function removes a virtual host from the catalog.

Input parameters

HOST_DEFINITION

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LOOP ABEND CATALOG_DELETE_FAILURE

WBRP gate, RECOVER_DEFINITIONS function

The RECOVER_DEFINITIONS function recovers previously installed definitions from the global catalog on a CICS warm start. URIMAP definitions and virtual hosts are recovered.

Input parameters

None

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	CATALOG_BROWSE_FAILURE CATALOG_PURGE_FAILURE LOGIC_ERROR WAIT_PHASE_FAILURE ABEND

WBSR gate, RECEIVE function

The RECEIVE function receives an HTTP Request off a socket, and parses it in order to determine what to do with it.

Input parameters

INITIAL_RECEIVE
Indicates whether this is the first receive issued by the caller, Can be set any of these values:

YES|NO

Output parameters

TOKEN
Token uniquely identifying the WebRequestBlock associated with this HTTP request.

ATTACH_TRANSID
Transaction ID of Web alias transaction to be attached to continue processing the HTTP request.

FAILING_PROGRAM
Name of program which returned an error in the course of receiving the HTTP request.

CONNECTION_PERSIST
Indicates whether the HTTP Request included the HTTP 1.0 Keepalive header. Can be set to any of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_ANALYZER_SPECIFIED ANALYZER_LINK_ERROR ANALYZER_ERROR ANALYZER_DATALENG_ERROR HDR_LENGTH_ERROR RECEIVE_ERROR STORAGE_ERROR CONNECTION_CLOSED LOGIC_ERROR

WBSR gate, SEND function

The SEND function returns the response constructed following receipt of an HTTP request.

Input parameters

TOKEN
Token identifying WebRequestBlock with which this SEND is associated

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	STORAGE_ERROR SEND_ERROR CONNECTION_CLOSED LOGIC_ERROR

WBSR gate, SEND_STATIC_RESPONSE function

The SEND_STATIC_RESPONSE function returns a static response specified by a URIMAP definition following receipt of an HTTP request.

Input parameters

None

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	

WBUR gate, URIMAP attributes

This list of URIMAP parameters is common to several functions on the WBUR gate.

Parameters

[SCHEME]
[USAGE]
[STATUS]
[ANALYZER]
[TRANSACTION]
[CONVERTER]
[PROGRAM]
[PIPELINE_NAME]
[WEBSERVICE_NAME]
[USERID]
[MEDIATYPE]
[CHARACTERSET]
[HOSTCODEPAGE]
[TEMPLATENAME]
[CERTIFICATE_LABEL]
[CIPHER_COUNT CIPHER_SUITES]
[REDIRECTION_TYPE]

WBUR gate, INITIALIZE_URIMAPS function

The INITIALIZE_URIMAPS function initializes the Web domain state required by the URIMAP support.

Input parameters

None

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION.

WBUR gate, ADD_REPLACE_URIMAP function

The ADD_REPLACE_URIMAP function adds or replaces a URIMAP definition into the Web domain.

Input parameters

URIMAP
HOST
PATH
[HFSFILE]
[REDIRECTION_LOCATION]
URIMAP attributes
see "WBUR gate, URIMAP attributes"

[TCPIPSERVICE]

Output parameters

[DUPLICATE_URIMAP]
RESPONSE
is the domain's response to the call. It can have any of these values:
`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION.

WBUR gate, DELETE_URIMAP function

The DELETE_URIMAP function deletes a URIMAP definition from the Web domain.

Input parameters

URIMAP

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION.

WBUR gate, LOCATE_URIMAP function

The LOCATE_URIMAP function is used to locate a URIMAP definition associated with a specified HOST and PATH.

Input parameters

HOST
PATH
[HFSFILE]
[REDIRECTION_LOCATION]
[TCPIPSERVICE]

Output parameters

URIMAP

URIMAP attributes
see "WBUR gate, URIMAP attributes" on page 1268

[UME_TOKEN]

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION.

WBUR gate, INQUIRE_URIMAP function

The INQUIRE_URIMAP function is used to inquire on the attributes of a URIMAP resource.

Input parameters

URIMAP
[HOST]
[PATH]
[HFSFILE]
[REDIRECTION_LOCATION]

Output parameters

URIMAP attributes
see "WBUR gate, URIMAP attributes" on page 1268

[TCPIPSERVICE]

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION.

WBUR gate, SET_URIMAP function

The SET_URIMAP function is used to set the attributes of a URIMAP resource.

Input parameters

URIMAP
[HOST]
[PATH]
[HFSFILE]

[REDIRECTION_LOCATION]
[TCPIPSERVICE]
URIMAP attributes see “WBUR gate, URIMAP attributes” on page 1268

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values: `OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBUR gate, START_BROWSE_URIMAP function

The START_BROWSE_URIMAP function is used to begin a browse through the URIMAP resources in the Web domain.

Input parameters

None

Output parameters

BROWSE_TOKEN

RESPONSE is the domain’s response to the call. It can have any of these values: `OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBUR gate, GET_NEXT_URIMAP function

The GET_NEXT_URIMAP function is used to continue a browse through the URIMAP resources in the Web domain.

Input parameters

BROWSE_TOKEN
[HOST]
[PATH]
[HFSFILE]
[REDIRECTION_LOCATION]

Output parameters

URIMAP
[TCPIPSERVICE]
URIMAP attributes see “WBUR gate, URIMAP attributes” on page 1268

RESPONSE is the domain’s response to the call. It can have any of these values: `OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBUR gate, END_BROWSE_URIMAP function

The END_BROWSE_URIMAP function is used to end a browse through the URIMAP resources in the Web domain.

Input parameters

BROWSE_TOKEN

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values: `OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBUR gate, INQUIRE_HOST function

The INQUIRE_HOST function is used to inquire on the attributes of a virtual host.

Input parameters
HOST
[TCPIPSERVICE]

Output parameters
STATUS

RESPONSE is the domain's response to the call. It can have any of these values:
`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBUR gate, SET_HOST function

The SET_HOST function is used to set the attributes of a virtual host.

Input parameters
HOST
[TCPIPSERVICE]
STATUS

Output parameters
RESPONSE is the domain's response to the call. It can have any of these values:
`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBUR gate, START_BROWSE_HOST function

The START_BROWSE_HOST function is used to begin a browse through the virtual host names in the Web domain.

Input parameters
None

Output parameters
BROWSE_TOKEN

RESPONSE is the domain's response to the call. It can have any of these values:
`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBUR gate, GET_NEXT_HOST function

The GET_NEXT_HOST function is used to continue a browse through the virtual host names in the Web domain.

Input parameters
BROWSE_TOKEN
HOST

Output parameters
TCPIPSERVICE
STATUS

RESPONSE is the domain's response to the call. It can have any of these values:
`OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION.

WBUR gate, END_BROWSE_HOST function

The END_BROWSE_HOST function is used to end a browse of the virtual host names in the Web domain.

| Input parameters
| **BROWSE_TOKEN**

| Output parameters

RESPONSE	is the domain's response to the call. It can have any of these values: OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED
[REASON]	is returned when RESPONSE is EXCEPTION.

Web domain's generic gates

Table 115 summarizes the Web domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 115. Web domain's generic gates

Gate	Trace	Function	Format
DMDM	WB 0100 WB 0101	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
XMAC	WB 0600 WB 0601	INIT_XM_CLIENT BIND_XM_CLIENT TRANSACTION_HANG RELEASE_XM_CLIENT	XMAC

For descriptions of these functions and their input and output parameters, refer to the Sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669

Format XMAC—Chapter 113, "Transaction manager domain (XM)," on page 1275

In initialization, quiesce, and termination processing, the Web domain performs only internal routines.

Modules

Module	Function
DFHWBDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHWBAP	Handles the following requests START_BROWSE READ_NEXT END_BROWSE GET_MESSAGE_BODY GET_HTTP_RESPONSE SEND_RESPONSE READ WRITE_HEADER INQUIRE

Module	Function
DFHWBAPF	Handles forms processing for: START_BROWSE READ_NEXT END_BROWSE READ
DFHWBSR	Handles the following requests SEND RECEIVE SEND_STATIC_RESPONSE
DFHWBXM	Handles the following requests INIT_XM_CLIENT BIND_XM_CLIENT TRANSACTION_HANG RELEASE_XM_CLIENT
DFHWBQM	Domain subroutine which writes Web data to TS. Handles the following requests: PUT_QUEUE GET_QUEUE DELETE_QUEUE GET_TOKEN
DFHWBCL	Functions for HTTP client processing.
DFHWBUR	Functions for handling the URIMAP resource, including virtual hosts.
DFHWBRP	Web domain recovery program.
DFHWBRQ	Shared code and data areas for Web domain functions.

Exits

Two global user exit points are provided in CICS Web support for HTTP client requests:

XWBOPEN, HTTP client open exit
XWBOPEN is called during processing of an EXEC CICS WEB OPEN command, which is used by an application program to open a connection with a server. It is designed for use to specify proxy servers that should be used for HTTP requests by CICS as an HTTP client, and to apply a security policy to the host name specified for those requests.

XWBSNDO, HTTP client send exit
XWBSNDO is called during processing of an EXEC CICS WEB SEND or EXEC CICS WEB CONVERSE command. It is designed for use to specify a security policy for HTTP requests, in particular for the path component of the request.

The exits are described in the *CICS Internet Guide*.

Trace

The point IDs for the Web domain are of the form WB xxxx; the corresponding trace levels are WB 1, WB 2, and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 113. Transaction manager domain (XM)

The transaction manager domain (also sometimes known simply as “transaction manager”) provides transaction-related services to:

- Create tasks
- Terminates tasks
- Purge tasks
- Inquire on tasks
- Manage transaction definitions
- Manage tranclass definitions.

The transaction manager domain also provides a transaction environment to enable other CICS components to implement transaction-related services.

Transaction manager domain’s specific gates

Table 116 summarizes the transaction manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 116. Transaction manager domain’s specific gates

Gate	Trace	Function	XPI
XMAT	XM 1101 XM 1102	ATTACH	NO
XMBD	XM 0501 XM 0502	START_BROWSE_TRANDEF GET_NEXT_TRANDEF END_BROWSE_TRANDEF	NO NO NO
XMCL	XM 0A01 XM 0A02	ADD_REPLACE_TCLASS ADD_TCLASS INQUIRE_TCLASS INQUIRE_ALL_TCLASSES SET_TCLASS DELETE_TCLASS START_BROWSE_TCLASS GET_NEXT_TCLASS END_BROWSE_TCLASS REGISTER_TCLASS_USAGE DEREGISTER_TCLASS_USAGE LOCATE_AND_LOCK_TCLASS UNLOCK_TCLASS	NO NO YES NO NO NO NO NO NO NO NO NO NO
XMDD	XM 0601 XM 0602	DELETE_TRANDEF	NO
XMER	XM 1204 XM 1205	SET_DEFERRED_MESSAGE INQUIRE_DEFERRED_MESSAGE SET_DEFERRED_ABEND INQUIRE_DEFERRED_ABEND REPORT_MESSAGE ABEND_TRANSACTION	NO NO NO NO NO NO
XMFD	XM 0701 XM 0702	FIND_PROFILE	NO

Table 116. Transaction manager domain's specific gates (continued)

Gate	Trace	Function	XPI
XMIQ	XM 1001 XM 1002	INQUIRE_TRANSACTION SET_TRANSACTION START_BROWSE_TRANSACTION GET_NEXT_TRANSACTION END_BROWSE_TRANSACTION START_BROWSE_TXN_TOKEN GET_NEXT_TXN_TOKEN END_BROWSE_TXN_TOKEN INQUIRE_TRANSACTION_TOKEN SET_TRANSACTION_TOKEN PURGE_TRANSACTION	YES YES NO NO NO NO NO NO
XMLD	XM 0401 XM 0402	LOCATE_AND_LOCK_TRANDEF UNLOCK_TRANDEF	NO NO
XMSR	XM 0801 XM 0802	INQUIRE_MXT SET_MXT INQUIRE_DTRTRAN SET_DTRTRAN	YES NO YES NO
XMXD	XM 0201 XM 0202	ADD_REPLACE_TRANDEF SET_TRANDEF INQUIRE_TRANDEF INQUIRE_REMOTE_TRANDEF	NO NO YES NO
XMXE	XM 1401 XM 1402	GET_TXN_ENVIRONMENT FREE_TXN_ENVIRONMENT	NO NO

XMAT gate, ATTACH function

The ATTACH function of the XMAT gate is used to attach a new transaction.

Input parameters

TRANSACTION_ID
The transaction identifier to attach.

TPNAME
Alternative means of specifying the transaction identifier to attach.

[ATTACH_PARMS]
Parameters to be passed to the attached transaction.

[PRIORITY]
Combined user and terminal priority to be added to that of the transaction definition to determine the total priority of the attached transaction.

[TOTAL_PRIORITY]
The overriding priority to be associated with the attached transaction.

FACILITY_TYPE
The type of principal facility to be associated with the attached transaction. It can have any of these values:

NONE|TERMINAL|TD|START

START_CODE
Indicates the reason for the attach It can have any of these values:

C|T|TT|QD|S|SD|SZ|DF

[TF_TOKEN]
Token identifying a terminal to be associated with the transaction.

[IC_TOKEN]
Token identifying a START request to be associated with the transaction.

[TD_TOKEN]
Token identifying a TDQ to be associated with the transaction.

[US_TOKEN]
Token identifying a user to be associated with the transaction.

[SYSTEM_ATTACH]
Indicates whether the transaction should be attached as a system transaction. It can have either of these values:

YES|NO

[SUSPEND] Indicates whether the attacher is willing to suspend during the attach. It can have either of these values:

YES|NO

RETURN_NOT_FOUND
Indicates whether the attacher wishes to receive the NOT_FOUND exception. Default is to attach CSAC in place of the requested transaction. It can have either of these values:

YES|NO

[RESTART_COUNT]
If the attach is for a restarted transaction then this count indicates the number of this restart attempt.

Output parameters

[TRANSACTION_TOKEN]
Is the token identifying the newly attached transaction.

[TRANNUM] Is the transaction number assigned to the newly attached transaction.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	NOT_FOUND DISABLED INSUFFICIENT_STORAGE NOT_ENABLED_FOR_SHUTDOWN
INVALID	INVALID_FUNCTION

XMBD gate, START_BROWSE_TRANDEF function

The START_BROWSE_TRANDEF function of the XMBD gate is used to initiate a browse of installed transaction definitions.

Input parameters

[START_AT] Identifies a transaction identifier that the browse is to start at.

Output parameters

BROWSE_TOKEN Token identifying this transaction definition browse.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	INVALID_FUNCTION

XMBD gate, GET_NEXT_TRANDEF function

The GET_NEXT_TRANDEF function of the XMBD gate is used to return information about the next transaction definition in the browse.

Input parameters

BROWSE_TOKEN Token identifying this browse of the transaction definitions.

Output parameters

[TRANSACTION_ID]
Transaction identifier

[INITIAL_PROGRAM]
Initial program of transaction.

[PROFILE_NAME]
Profile of transaction.

[TWASIZE]
Size of Transaction Work Area.

[TRAN_PRIORITY]
Transaction priority

[STATUS]
The status of the transaction. It can have either of these values:

ENABLED|DISABLED

[PARTITIONSET]
The partitionset defined for the transaction. It can have any of these values:

NONE|NAMED|KEEP|OWN

[PARTITIONSET_NAME]
The name of the user defined partitionset used by the transaction.

[TASKDATAKEY]
The storage key that task-lifetime storage is allocated in. It can have either of these values:

CICS|USER

[TASKDATALOC]
The location of task-lifetime storage. It can have either of these values:

BELOW|ANY

[STORAGE_CLEAR]
Whether task-lifetime storage is to be cleared before it is freemained. It can have either of these values:

YES|NO

[SYSTEM_RUNAWAY]
Whether the transaction uses the default system runaway limit. It can have either of these values:

YES|NO

[RUNAWAY_LIMIT]
The runaway limit associated with the transaction.

[DYNAMIC]
Whether the transaction is defined to be dynamic. It can have either of these values:

YES|NO

[LOCAL_QUEUING]
Whether the transaction is eligible to queue locally when it is started on the remote system. It can have either of these values:

YES|NO

[REMOTE]
Whether the transaction is remote. It can have either of these values:

YES|NO

[REMOTE_SYSTEM]
The system that a remote transaction is to be routed to.

[REMOTE_NAME]
The name of a remote transaction on the remote system.

[TRAN_ROUTING_PROFILE]
Profile to be used to route a remote transaction to a remote system.

[TCLASS]
Whether the transaction belongs to a tclass. It can have either of these values:

YES|NO

[TCLASS_NAME]
The name of the tclass that the transaction belongs to.

[INDOUBT]
The action to take if work performed by the transaction becomes indoubt. It can have any of these values:

BACKOUT|COMMIT|WAIT

[RESTART] Whether the transaction is restartable. It can have either of these values:

YES|NO

[SPURGE] Whether the transaction is system-purgeable. It can have either of these values:

YES|NO

[DTIMEOUT] The deadlock timeout value for the transaction.

[TPURGE] Whether the transaction can be purged after a terminal error. It can have either of these values:

YES|NO

[DUMP] Whether transaction dumps are to be taken. It can have either of these values:

YES|NO

[TRACE] The level of tracing associated with the transaction. It can have any of these values:

STANDARD|SPECIAL|SUPPRESSED

[SHUTDOWN] Whether the transaction can be run during shutdown. It can have either of these values:

ENABLED|DISABLED

[RESSEC] Whether resource security checking is active. It can have either of these values:

YES|NO

[CMDSEC] Whether command security checking is active. It can have either of these values:

YES|NO

[STORAGE_FREEZE]
Whether storage freeze is on for the transaction. It can have either of these values:

YES|NO

[ISOLATE] Whether the transaction runs in its own subspace. It can have either of these values:

YES|NO

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
EXCEPTION	BROWSE_END_TRANDEF
INVALID	INVALID_BROWSE_TOKEN INVALID_FUNCTION

XMBD gate, END_BROWSE_TRANDEF function

The END_BROWSE_TRANDEF function of the XMBD gate is used to terminate a browse of installed transaction definitions.

Input parameters

BROWSE_TOKEN Token identifying this transaction definition browse.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP

RESPONSE	Possible REASON values
INVALID	INVALID_BROWSE_TOKEN INVALID_FUNCTION

XMCL gate, ADD_REPLACE_TCLASS function

The ADD_REPLACE_TCLASS function of the XMCL gate is used to install a tclass definition.

Input parameters

TCLASS_NAME The name of the tclass.

MAX_ACTIVE The max-active limit of the tclass.

[PURGE_THRESHOLD]
The purge-threshold limit of the tclass.

Output parameters

[TCLASS_TOKEN]
Token identifying the tclass.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
EXCEPTION	INVALID_TCLASS_NAME INVALID_MAX_ACTIVE INVALID_PURGE_THRESHOLD
INVALID	INVALID_FUNCTION

XMCL gate, ADD_TCLASS function

The ADD_TCLASS function of the XMCL gate is used to add an internal tclass definition.

Input parameters

[TCLASS_NAME]
The name of the tclass.

MAX_ACTIVE The max-active limit of the tclass.

[PURGE_THRESHOLD]
The purge-threshold limit of the tclass.

Output parameters

TCLASS_TOKEN Token identifying the tclass.

RESPONSE is the domain's response to the call. Possible values are:
OK|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. It can have any of these values:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
EXCEPTION	DUPLICATE_TCLASS_NAME INVALID_TCLASS_NAME INVALID_MAX_ACTIVE INVALID_PURGE_THRESHOLD

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION

XMCL gate, INQUIRE_TCLASS function

The INQUIRE_TCLASS function of the XMCL gate is used to inquire upon a tclass.

Input parameters

INQ_TCLASS_NAME
The name of the tclass being inquired upon.

TCLASS_TOKEN
Token identifying tclass being inquired upon.

Output parameters

[TCLASS_NAME]
The name of the tclass.

[MAX_ACTIVE]
The max-active limit of the tclass.

[PURGE_THRESHOLD]
The purge-threshold limit of the tclass.

[CURRENT_ACTIVE]
The number of active transactions in the tclass.

[CURRENT_QUEUED]
The number of queuing transactions in the tclass.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
EXCEPTION	UNKNOWN_TCLASS
INVALID	INVALID_TCLASS_TOKEN INVALID_FUNCTION

XMCL gate, INQUIRE_ALL_TCLASSES function

The INQUIRE_ALL_TCLASSES function of the XMCL gate is used to inquire about the current state of all the tclasses in the system.

Input parameters

None.

Output parameters

[TOTAL_ACTIVE]
The number of transactions active in a tclass.

[TOTAL_QUEUED]
The number of transactions queueing for a tclass.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
EXCEPTION	UNKNOWN_TCLASS
LOGIC_ERROR	INVALID_FUNCTION

XMCL gate, SET_TCLASS function

The SET_TCLASS function of the XMCL gate is used to modify a tclass definition.

Input parameters

TCLASS_NAME The name of the tclass to be changed.

TCLASS_TOKEN Token identifying tclass to be changed.

[MAX_ACTIVE] The max-active limit of the tclass.

[PURGE_THRESHOLD]
The purge-threshold limit of the tclass.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
EXCEPTION	UNKNOWN_TCLASS INVALID_MAX_ACTIVE INVALID_PURGE_THRESHOLD
INVALID	INVALID_TCLASS_TOKEN INVALID_FUNCTION

XMCL gate, DELETE_TCLASS function

The DELETE_TCLASS function of the XMCL gate is used to discard an installed tclass definition.

Input parameters

TCLASS_NAME The name of the tclass to be deleted.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
EXCEPTION	UNKNOWN_TCLASS TCLASS_BUSY
INVALID	INVALID_FUNCTION

XMCL gate, START_BROWSE_TCLASS function

The START_BROWSE_TCLASS function of the XMCL gate is used to initiate a browse of installed tclass definitions.

Input parameters

[START_AT] Identifies a tclass that the browse is to start at.

Output parameters

BROWSE_TOKEN Token identifying this tclass browse.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	INVALID_FUNCTION

XMCL gate, GET_NEXT_TCLASS function

The GET_NEXT_TCLASS function of the XMCL gate is used to return information about the next tclass definition in the browse.

Input parameters

BROWSE_TOKEN Token identifying this browse of the tclass definitions.

Output parameters

[TCLASS_NAME]
The name of the tclass.

[MAX_ACTIVE] The max-active limit of the tclass.

[PURGE_THRESHOLD]
The purge-threshold limit of the tclass.

[CURRENT_ACTIVE]
The number of active transactions in the tclass.

[CURRENT_QUEUED]
The number of queuing transactions in the tclass.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
EXCEPTION	BROWSE_END_TCLASS
INVALID	INVALID_BROWSE_TOKEN INVALID_FUNCTION

XMCL gate, END_BROWSE_TCLASS function

The END_BROWSE_TCLASS function of the XMCL gate is used to terminate a browse of installed tclass definitions.

Input parameters

BROWSE_TOKEN Token identifying this tclass browse.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	INVALID_BROWSE_TOKEN INVALID_FUNCTION

XMCL gate, REGISTER_TCLASS_USAGE function

The REGISTER_TCLASS_USAGE function of the XMCL gate is used to register usage of a tclass by a transaction definition.

Input parameters

TCLASS_NAME The name of the tclass that is being used.

UNKNOWN_ACTION

Specifies the action to perform if the tclass hasn't been installed by the user: It can have either of these values:

CREATE|ERROR

Output parameters

TCLASS_TOKEN Token identifying tclass.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
EXCEPTION	UNKNOWN_TCLASS
INVALID	INVALID_FUNCTION

XMCL gate, DEREGISTER_TCLASS_USAGE function

The DEREGISTER_TCLASS_USAGE function of the XMCL gate is used to deregister usage of a tclass by a transaction definition.

Input parameters

TCLASS_TOKEN Token identifying tclass that is no longer being used.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_TCLASS_TOKEN, NOT_IN_USE, INVALID_FUNCTION

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR, ABEND, LOOP

XMCL gate, LOCATE_AND_LOCK_TCLASS function

The LOCATE_AND_LOCK_TCLASS function of the XMCL gate is used to locate a named tclass and lock it against delete.

Input parameters

TCLASS_NAME Name of tclass to be located.

Output parameters

TCLASS_TOKEN Token identifying tclass.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
EXCEPTION	UNKNOWN_TCLASS
INVALID	INVALID_FUNCTION

XMCL gate, UNLOCK_TCLASS function

The UNLOCK_TCLASS function of the XMCL gate is used to unlock a previously locked tclass.

Input parameters

TCLASS_TOKEN Token identifying tclass to be unlocked.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR, ABEND, LOOP
INVALID	INVALID_TCLASS_TOKEN, NOT_LOCKED, INVALID_FUNCTION

XMDD gate, DELETE_TRANDEF function

The DELETE_TRANDEF function of the XMDD gate is used to discard an installed transaction definition.

Input parameters

TRANSACTION_ID

The name of the transaction to be deleted.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
EXCEPTION	UNKNOWN_TRANSACTION_ID ICE_PENDING AID_PENDING SIT_PARAMETER
INVALID	INVALID_FUNCTION

XMER gate, SET_DEFERRED_MESSAGE function

The SET_DEFERRED_MESSAGE function of the XMER gate is used to store a message to be issued if the attach of a transaction fails.

Input parameters

MESSAGE
The message that is to be issued.

[TRANSACTION_TOKEN]
Optional token to identify the transaction that the message is to be sent to. Defaults to the current transaction.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	MESSAGE_ALREADY_SET DEFERRED_ABEND_ALREADY_SET INVALID_TRANSACTION_TOKEN
INVALID	INVALID_FUNCTION

XMER gate, INQUIRE_DEFERRED_MESSAGE function

The INQUIRE_DEFERRED_MESSAGE function of the XMER gate is used to retrieve the message that is to be issued which will indicate the cause of a transaction attach failure.

Output parameters

MESSAGE
The message that is to be issued.

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	MESSAGE_NOT_FOUND
INVALID	INVALID_FUNCTION

XMER gate, SET_DEFERRED_ABEND function

The SET_DEFERRED_ABEND function of the XMER gate is used to schedule an abend to be issued if the attach of a transaction fails.

Input parameters

DEFERRED_ABEND_CODE
The abend code that is to be used.

[TRANSACTION_DUMP]
Indicates whether a transaction dump is to be taken for the abend. It can have either of these values:

YES|NO

[TRANSACTION_TOKEN]
Optional token to identify the transaction that is to be abended. Defaults to the current transaction.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	MESSAGE_ALREADY_SET DEFERRED_ABEND_ALREADY_SET INVALID_TRANSACTION_TOKEN
INVALID	INVALID_ABEND_CODE INVALID_FUNCTION

XMER gate, INQUIRE_DEFERRED_ABEND function

The INQUIRE_DEFERRED_ABEND function of the XMER gate is used to retrieve the abend that is to be issued for the transaction whose attach has failed.

Output parameters

DEFERRED_ABEND_CODE
The abend code.

[TRANSACTION_DUMP]
Indicates whether a transaction dump is to be taken for the abend. It can have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	DEFERRED_ABEND_NOT_FOUND
INVALID	INVALID_FUNCTION

XMER gate, REPORT_MESSAGE function

The REPORT_MESSAGE function of the XMER gate is used send a deferred message if the attach of a transaction has failed.

Input parameters

MESSAGE
The message that is to be sent.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	TRANSACTION_ABEND
INVALID	INVALID_FUNCTION

XMER gate, ABEND_TRANSACTION function

The ABEND_TRANSACTION function of the XMER gate is used abend a transaction whose attach has failed.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION

XMFD gate, FIND_PROFILE function

The FIND_PROFILE function of the XMFD gate is used to check whether the given profile is in use by a transaction definition.

Input parameters

PROFILE_NAME
The profile that is to be found.

Output parameters

[TRANSACTION_ID]
The name of a transaction definition that is using the profile.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
EXCEPTION	PROFILE_NOT_FOUND
INVALID	INVALID_FUNCTION

XMIQ gate, INQUIRE_TRANSACTION function

The INQUIRE_TRANSACTION function of the XMIQ gate is used to inquire upon a particular transaction.

Input parameters

[TRANSACTION_NUMBER]
The number of the transaction being inquired upon.

[TRANSACTION_TOKEN]
Or the token representing the transaction being inquired upon.

If neither TRANSACTION_NUMBER or TRANSACTION_TOKEN are specified the current transaction is assumed.

[ATTACH_PARMS]
Specified if the parameter area passed on the transaction. attach are to be returned.

Output parameters

[ABEND_CODE] The abend code if the transaction is terminating abnormally.

[ABEND_IN_PROGRESS]
Indicates whether the transaction is in the process of terminating abnormally. It can have either of these values:

YES|NO

[CICS_UOW_ID]
The CICS Unit Of Work Identifier associated with the transaction.

[CMDSEC] Whether command security checking is active. It can have either of these values:

YES|NO

[DTIMEOUT] The deadlock timeout value for the transaction.

[DUMP] Whether transaction dumps are to be taken for the transaction. It can have either of these values:

YES|NO

[DYNAMIC] Whether the transaction is dynamic. It can have either of these values:

YES|NO

[FACILITY_NAME]
The name of the principal facility associated with the transaction.

[FACILITY_TYPE]
The type of the principal facility associated with the transaction. It can have either of these values:

NONE|TERMINAL|TD|START

[INDOUBT] The action to take if work performed by the transaction becomes indoubt. It can have any of these values:

BACKOUT|COMMIT|WAIT

[INITIAL_PROGRAM]
The initial program to linked to when the transaction started.

[ISOLATE] Whether the transaction runs in its own subspace. It can have either of these values:

YES|NO

[LOCAL_QUEUING]
Whether the transaction is eligible to queue locally if it is started on the remote system. It can have either of these values:

YES|NO

[NETNAME] The network name of a terminal principal facility.

[ORIGINAL_TRANSACTION_ID]
The transid that was used to attach the transaction.

[OUT_TRANSACTION_TOKEN]
The token that represents this transaction.

[PHASE] The phase of the transaction. It can have one of these values:

PRE_INIT|INIT|BIND|TERM

[PROFILE_NAME]
The profile of the transaction.

[REMOTE] Whether the transaction is remote. It can have either of these values:

YES|NO

[REMOTE_NAME]
The name of a remote transaction on the remote system.

[REMOTE_SYSTEM]
The system that a remote transaction is to be routed to.

[RESOURCE_NAME]
The name of a resource that a suspended transaction is waiting for.

[RESOURCE_TYPE]
The type of resource that a suspended transaction is waiting for.

[RESSEC]
Whether resource security checking is active for the transaction. It can have either of these values:

`YES|NO`

[RESTART]
Whether the transaction is restartable. It can have either of these values:

`YES|NO`

[RESTART_COUNT]
Contains the number of times this transaction instance has been restarted.

[RUNAWAY_LIMIT]
The runaway limit associated with the transaction.

[SPURGE]
Whether the transaction is system-purgeable. It can have either of these values:

`YES|NO`

START_CODE
Indicates the reason for the attach of the transaction. It can have any of these values:

`C|T|TT|QD|S|SD|SZ|DF`

[STATUS]
The status of the transaction. It can have either of these values:

`ENABLED|DISABLED`

[STORAGE_CLEAR]
Whether task-lifetime storage will be cleared before it is freemained. It can have either of these values:

`YES|NO`

[SUSPEND_TIME]
Contains the length of time that the transaction has currently been suspended for.

[SYSTEM_TRANSACTION]
Whether the transaction has been attached by CICS. It can have either of these values:

`YES|NO`

[TASK_PRIORITY]
The combined priority of the transaction.

[TASKDATAKEY]
The storage key that task-lifetime storage is allocated in. It can have either of these values:

`CICS|USER`

[TASKDATALOC]
The location of task-lifetime storage. It can have either of these values:

`BELOW|ANY`

[TCLASS]
Whether the transaction belongs to a tclass. It can have either of these values:

`YES|NO`

[TCLASS_NAME]
The name of the tclass that the transaction belongs to.

[TPURGE]
Whether the transaction can be purged after a terminal error. It can have either of these values:

`YES|NO`

[TRACE]
The level of tracing associated with the transaction. It can have any of these values:

`STANDARD|SPECIAL|SUPPRESSED`

[TRAN_PRIORITY]
The priority of the transaction definition used to attach the transaction.

[TRAN_ROUTING_PROFILE]
Profile used to route the transaction to a remote system.

[TRANNUM] The transaction number of the transaction.

[TRANSACTION_ID]
The transaction identifier associated with the transaction.

[TWASIZE] Size of Transaction Work Area associated with the transaction.

[USERID] The userid of the user associated with the transaction.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	NO_TRANSACTION_ENVIRONMENT BUFFER_TOO_SMALL UNKNOWN_TRANSACTION_NUMBER INVALID_TRANSACTION_TOKEN
INVALID	INVALID_FUNCTION

XMIQ gate, SET_TRANSACTION function

The SET_TRANSACTION function of the XMIQ gate is used to change some attributes associated with a particular transaction.

Input parameters

[TRANSACTION_NUMBER]
The number of the transaction being inquired upon.

[TRANSACTION_TOKEN]
Or the token representing the transaction being inquired upon.

If neither TRANSACTION_NUMBER or TRANSACTION_TOKEN are specified the current transaction is assumed.

[ABEND_CODE] The abend code if the transaction is terminating abnormally.

[ABEND_IN_PROGRESS]
Whether the transaction is in the process of terminating abnormally. It can have either of these values:

YES|NO

[FACILITY_TYPE]
The type of the principal facility associated with the transaction. It can have either of these values:

NONE|TERMINAL|TD|START

START_CODE The reason for the attach of the transaction. It can have any of these values:

C|T|TT|QD|S|SD|SZ|DF

[STORAGE_VIOLATIONS]
Set to indicate that the transaction has suffered a storage violation.

[TASK_PRIORITY]
The combined priority of the transaction.

[TCLASS_NAME]
The name of the tclass that the transaction belongs to.

Reserved name DFHTCL00 is used to change a transaction so that it no longer belongs to a tclass.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	NO_TRANSACTION_ENVIRONMENT UNKNOWN_TCLASS UNKNOWN_TRANSACTION_NUMBER INVALID_TRANSACTION_TOKEN
INVALID	INVALID_FUNCTION

XMIQ gate, START_BROWSE_TRANSACTION function

The START_BROWSE_TRANSACTION function of the XMIQ gate is used to initiate a browse of all transactions in the system.

Output parameters

BROWSE_TOKEN Token identifying this transaction browse.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION

XMIQ gate, GET_NEXT_TRANSACTION function

The GET_NEXT_TRANSACTION function of the XMIQ gate is used to inquire upon the next transaction in a transaction browse.

Input parameters

BROWSE_TOKEN The token identifying this transaction browse.

[ATTACH_PARMS]
Specified if the parameter area passed on the transaction. attach is to be returned.

Output parameters

[ABEND_CODE] The abend code if the transaction is terminating abnormally.

[ABEND_IN_PROGRESS]
Indicates whether the transaction is in the process of terminating abnormally. It can have either of these values:

YES|NO

[CICS_UOW_ID]
The CICS Unit Of Work Identifier associated with the transaction.

[CMDSEC] Whether command security checking is active. It can have either of these values:

YES|NO

[DTIMEOUT] The deadlock timeout value for the transaction.

[DUMP] Whether transaction dumps are to be taken for the transaction. It can have either of these values:

YES|NO

[DYNAMIC] Whether the transaction is dynamic. It can have either of these values:

YES|NO

[FACILITY_NAME]
The name of the principal facility associated with the transaction.

[FACILITY_TYPE]
The type of the principal facility associated with the transaction. It can have either of these values:

NONE|TERMINAL|TD|START

[INDOUBT]
The action to take if work performed by the transaction becomes indoubt. It can have any of these values:

BACKOUT|COMMIT|WAIT

[INITIAL_PROGRAM]
The initial program to linked to when the transaction started.

[ISOLATE]
Whether the transaction runs in its own subspace. It can have either of these values:

YES|NO

[LOCAL_QUEUING]
Whether the transaction is eligible to queue locally if it is started on the remote system. It can have either of these values:

YES|NO

[NETNAME]
The network name of a terminal principal facility.

[ORIGINAL_TRANSACTION_ID]
The transid that was used to attach the transaction.

[OUT_TRANSACTION_TOKEN]
The token that represents this transaction.

[PHASE]
The phase of the transaction. It can have one of these values:

PRE_INIT|INIT|BIND|TERM

[PROFILE_NAME]
The profile of the transaction.

[REMOTE]
Whether the transaction is remote. It can have either of these values:

YES|NO

[REMOTE_NAME]
The name of a remote transaction on the remote system.

[REMOTE_SYSTEM]
The system that a remote transaction is to be routed to.

[RESOURCE_NAME]
The name of a resource that a suspended transaction is waiting for.

[RESOURCE_TYPE]
The type of resource that a suspended transaction is waiting for.

[RESSEC]
Whether resource security checking is active for the transaction. It can have either of these values:

YES|NO

[RESTART]
Whether the transaction is restartable. It can have either of these values:

YES|NO

[RESTART_COUNT]
Contains the number of times this transaction instance has been restarted.

[RUNAWAY_LIMIT]
The runaway limit associated with the transaction.

[SPURGE]
Whether the transaction is system-purgeable. It can have either of these values:

YES|NO

START_CODE
Indicates the reason for the attach of the transaction. It can have any of these values:

C|T|TT|QD|S|SD|SZ|DF

[STATUS]
The status of the transaction. It can have either of these values:

ENABLED|DISABLED

[STORAGE_CLEAR]
Whether task-lifetime storage will be cleared before it is freemained. It can have either of these values:

YES|NO

[SUSPEND_TIME]
Contains the length of time that the transaction has currently been suspended for.

[SYSTEM_TRANSACTION]
Whether the transaction has been attached by CICS. It can have either of these values:

YES|NO

[TASK_PRIORITY]
The combined priority of the transaction.

[TASKDATAKEY]
The storage key that task-lifetime storage is allocated in. It can have either of these values:

CICS|USER

[TASKDATALOC]
The location of task-lifetime storage. It can have either of these values:

BELOW|ANY

[TCLASS]
Whether the transaction belongs to a tclass. It can have either of these values:

YES|NO

[TCLASS_NAME]
The name of the tclass that the transaction belongs to.

[TPURGE]
Whether the transaction can be purged after a terminal error. It can have either of these values:

YES|NO

[TRACE]
The level of tracing associated with the transaction. It can have any of these values:

STANDARD|SPECIAL|SUPPRESSED

[TRAN_PRIORITY]
The priority of the transaction definition used to attach the transaction.

[TRAN_ROUTING_PROFILE]
Profile used to route the transaction to a remote system.

[TRANNUM]
The transaction number of the transaction.

[TRANSACTION_ID]
The transaction identifier associated with the transaction.

[TWASIZE]
Size of Transaction Work Area associated with the transaction.

[USERID]
The userid of the user associated with the transaction.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END
DISASTER	ABEND LOOP
INVALID	INVALID_BROWSE_TOKEN INVALID_FUNCTION

XMIQ gate, END_BROWSE_TRANSACTION function

The END_BROWSE_TRANSACTION function of the XMIQ gate is used to terminate a browse of all transactions in the system.

Input parameters

BROWSE_TOKEN
Token identifying the transaction browse to be terminated.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK||DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_BROWSE_TOKEN INVALID_FUNCTION

XMIQ gate, START_BROWSE_TXN_TOKEN function

The START_BROWSE_TXN_TOKEN function of the XMIQ gate is used to initiate a browse of a particular components transaction token in all transactions in the system.

Input parameters

TOKEN_OWNER Identifies the particular transaction token that is to be browsed in the transactions. It can have any of these values:

AP|SM|TD|MN|PG|IC|XS|US|RM|TF

Output parameters

BROWSE_TOKEN Token identifying this transaction token browse.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND, LOOP
INVALID	INVALID_FUNCTION

XMIQ gate, GET_NEXT_TXN_TOKEN function

The GET_NEXT_TXN_TOKEN function of the XMIQ gate is used to return the transaction token associated with the next transaction in the system.

Input parameters

BROWSE_TOKEN Identifies this browse of the transaction tokens.

Output parameters

OWNERS_TOKEN The transaction token associated with the current transaction.

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	BROWSE_END
DISASTER	ABEND LOOP
INVALID	INVALID_BROWSE_TOKEN INVALID_FUNCTION

XMIQ gate, END_BROWSE_TXN_TOKEN function

The END_BROWSE_TXN_TOKEN function of the XMIQ gate is used to terminate a browse of transaction tokens.

Input parameters

BROWSE_TOKEN Token identifying the transaction token browse to be terminated.

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK||DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND, LOOP
INVALID	INVALID_BROWSE_TOKEN, INVALID_FUNCTION

XMIQ gate, INQUIRE_TRANSACTION_TOKEN function

The INQUIRE_TRANSACTION_TOKEN function of the XMIQ gate is used to return a particular transaction token associated with a particular transaction.

Input parameters

[TRANSACTION_TOKEN]
Token identifying the transaction being inquired upon.

If omitted defaults to the current transaction.

TOKEN_OWNER Identifies the particular transaction token that is to be returned. It can have any of these values:

AP|SM|TD|MN|PG|IC|XS|US|RM|TF

Output parameters

OWNERS_TOKEN The transaction token associated with the transaction.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_TRANSACTION_ENVIRONMENT
DISASTER	ABEND LOOP
INVALID	INVALID_BROWSE_TOKEN INVALID_FUNCTION

XMIQ gate, SET_TRANSACTION_TOKEN function

The SET_TRANSACTION_TOKEN function of the XMIQ gate is used to modify a particular transaction token associated with a particular transaction.

Input parameters

[TRANSACTION_TOKEN]
Token identifying the transaction in which the token is to be modified.

If omitted defaults to the current transaction.

TOKEN_OWNER Identifies the particular transaction token that is to be changed. It can have any of these values:

AP|SM|TD|MN|PG|IC|XS|US|RM|TF

OWNERS_TOKEN The new value for the transaction token.

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_TRANSACTION_ENVIRONMENT
DISASTER	ABEND LOOP
INVALID	INVALID_BROWSE_TOKEN INVALID_FUNCTION

XMIQ gate, PURGE_TRANSACTION function

The PURGE_TRANSACTION function of the XMIQ gate is used to purge a particular transaction in the system.

Input parameters

TRANSACTION_NUMBER
The number of the transaction to be purged.

TRANSACTION_TOKEN
Or the token representing the transaction to be purged.

PURGE_TYPE The type of purge that is to be attempted. It can have either of these values:

NORMAL|FORCE

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_TRANSACTION_NUMBER INVALID_TRANSACTION_TOKEN PURGE_DEFERRED TRANSACTION_INITIALIZING TRANSACTION_TERMINATING PURGE_SYSTEM_TRANSACTION PURGE_ABENDING_TRANSACTION SPURGE_PROTECTED PURGE_INHIBITED INVALID_STATE FORCEPURGE_NOT_ATTEMPTED
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION

XMLD gate, LOCATE_AND_LOCK_TRANDEF function

The LOCATE_AND_LOCK_TRANDEF function of the XMLD gate is used to locate a particular transaction definition instance.

Input parameters

TRANSACTION_ID
Transaction identifier to locate.

TPNAME Or alternatively a tpname alias of the transaction definition to locate.

[USE_DTRTRAN]
If the named transaction-id or tpname cannot be found then indicates whether the DTRTRAN, if installed, should be used instead. It can have either of these values:

YES|NO

Output parameters

TRANDEF_TOKEN
The token representing the returned transaction definition.

[PRIMARY_TRANSACTION_ID]
The primary transaction identifier of the returned transaction. definition.

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOT_FOUND
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	INVALID_TPNAME INVALID_FUNCTION

XMLD gate, UNLOCK_TRANDEF function

The UNLOCK_TRANDEF function of the XMLD gate is used to unlock a previously located transaction definition instance.

Input parameters

TRANDEF_TOKEN
Transaction definition instance to unlock.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	NOT_LOCKED INVALID_TOKEN INVALID_FUNCTION

XMSR gate, INQUIRE_MXT function

The INQUIRE_MXT function of the XMSR gate is used to inquire upon the state of MXT in the system.

Output parameters

[MXT_QUEUED]
The number of user transactions queued for MXT.

[TCLASS_QUEUED]
The number of transactions queued for tclass membership.

[CURRENT_ACTIVE]
The number of active user transactions.

[CURRENT_ACTIVE]
The number of user transactions queued on MXT.

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	INVALID_FUNCTION

XMSR gate, SET_MXT function

The SET_MXT function of the XMSR gate is used to change MXT in the system.

Input parameters

MXT_LIMIT The requested setting for MXT.

Output parameters

MXT_LIMIT_SET
The MXT limit that could be set.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INVALID_MXT_LIMIT LIMIT_TOO_HIGH
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	INVALID_FUNCTION

XMSR gate, INQUIRE_DTRTRAN function

The INQUIRE_DTRTRAN function of the XMSR gate returns the name of the dynamic transaction routing transaction.

Output parameters

DTRTRAN The name of the dynamic transaction routing transaction definition.

RESPONSE is the domain's response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	INVALID_FUNCTION

XMSR gate, SET_DTRTRAN function

The SET_DTRTRAN function of the XMSR gate changes the dynamic transaction routing transaction definition.

Input parameters

DTRTRAN The name of the dynamic transaction routing transaction definition.

Output parameters

RESPONSE is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	INVALID_FUNCTION

XMXD gate, ADD_REPLACE_TRANDEF function

The ADD_REPLACE_TRANDEF function of the XMXD gate is used to install a transaction definition.

Input parameters

TRANSACTION_ID
Name of transaction definition to install.

PROFILE_NAME Profile of transaction.

TRAN_PRIORITY
Transaction priority

[INITIAL_PROGRAM]
Initial program of transaction.

[TWASIZE] Size of Transaction Work Area.

[STATUS] The status of the transaction. It can have either of these values:

ENABLED|DISABLED

[PARTITIONSET]
The partitionset defined for the transaction. It can have any of these values:

NONE|NAMED|KEEP|OWN

[PARTITIONSET_NAME]
The name of the user defined partitionset used by the transaction.

[TASKDATAKEY]
The storage key that task-lifetime storage is allocated in. It can have either of these values:

CICS|USER

[TASKDATALOC]
The location of task-lifetime storage. It can have either of these values:

BELOW|ANY

[STORAGE_CLEAR]
Whether task-lifetime storage is to be cleared before it is freemained. It can have either of these values:

YES|NO

[SYSTEM_RUNAWAY]
Whether the transaction uses the default system runaway limit. It can have either of these values:

YES|NO

[RUNAWAY_LIMIT]
The runaway limit associated with the transaction.

[DYNAMIC] Whether the transaction is defined to be dynamic. It can have either of these values:

YES|NO

[LOCAL_QUEUING]
Whether the transaction is eligible to queue locally when it is started on the remote system. It can have either of these values:

YES|NO

[REMOTE_SYSTEM]
The system that a remote transaction is to be routed to.

[REMOTE_NAME]
The name of a remote transaction on the remote system.

[TRAN_ROUTING_PROFILE]
Profile to be used to route a remote transaction to a remote system.

[TCLASS]
Whether the transaction belongs to a tclass. It can have either of these values:

YES|NO

[TCLASS_NAME]
The name of the tclass that the transaction belongs to.

[INDOUBT]
The action to take if work performed by the transaction becomes indoubt. It can have any of these values:

BACKOUT|COMMIT|WAIT

[RESTART]
Whether the transaction is restartable. It can have either of these values:

YES|NO

[SPURGE]
Whether the transaction is system-purgeable. It can have either of these values:

YES|NO

[DTIMEOUT]
The deadlock timeout value for the transaction.

[TPURGE]
Whether the transaction can be purged after a terminal error. It can have either of these values:

YES|NO

[DUMP]
Whether transaction dumps are to be taken. It can have either of these values:

YES|NO

[TRACE]
The level of tracing associated with the transaction. It can have any of these values:

STANDARD|SPECIAL|SUPPRESSED

[SHUTDOWN]
Whether the transaction can be run during shutdown. It can have either of these values:

ENABLED|DISABLED

[RESSEC]
Whether resource security checking is active. It can have either of these values:

YES|NO

[CMDSEC]
Whether command security checking is active. It can have either of these values:

YES|NO

[STORAGE_FREEZE]
Whether storage freeze is on for the transaction. It can have either of these values:

YES|NO

[ISOLATE]
Whether the transaction runs in its own subspace. It can have either of these values:

YES|NO

[CATALOGUED_EXTERNALS]
Block of data specified as an alternative to the above parameters when a transaction definition is being installed from the catalog.

[ALIAS]
Alternative name for transaction definition.

[TASKREQ]
Alternative name for transaction definition so that it can be invoked by PF/PA key, light pen, etc.

[XTRANID]
Alternative name for transaction definition originally specified in hexadecimal notation.

[TPNAME]
Alternative name of transaction definition in form of a sixty four character transaction program name.

[SYSTEN_DEFINITION]
Whether the definition is being added on behalf of CICS or not. It can have either of these values:

YES|NO

Output parameters

[TRANDEF_TOKEN]
Token returned to represent the installed transaction. definition.

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	TWASIZE_INVALID RUNAWAY_LIMIT_INVALID TRANSACTION_ID_INVALID ALIAS_INVALID XTRANID_INVALID TASKREQ_INVALID TPNAME_INVALID RECOVERY_NOT_COMPLETE
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	INITIAL_PROGRAM_EXPECTED REMOTE_SYSTEM_EXPECTED REMOTE_NAME_EXPECTED RUNAWAY_LIMIT_EXPECTED TRAN_ROUTING_PROF_EXPECTED TCLASS_NAME_EXPECTED PARTITIONSET_NAME_EXPECTED INVALID_FUNCTION

XMXD gate, SET_TRANDEF function

The SET_TRANDEF function of the XMXD gate is used to modify transaction definition creating a new transaction. definition instance.

Input parameters

TRANSACTION_ID
Name of transaction definition to change.

[TRAN_PRIORITY]
Transaction priority.

[STATUS] The status of the transaction. It can have either of these values:

`ENABLED|DISABLED`

[SYSTEM_RUNAWAY]
Whether the transaction uses the default system runaway limit. It can have either of these values:

`YES|NO`

[RUNAWAY_LIMIT]
The runaway limit associated with the transaction.

[TCLASS] Whether the transaction belongs to a tclass. It can have either of these values:

`YES|NO`

[TCLASS_NAME]
The name of the tclass that the transaction belongs to.

[SPURGE] Whether the transaction is system-purgeable. It can have either of these values:

`YES|NO`

[DUMP] Whether transaction dumps are to be taken. It can have either of these values:

`YES|NO`

[TRACE] The level of tracing associated with the transaction. It can have any of these values:

`STANDARD|SPECIAL|SUPPRESSED`

[SHUTDOWN] Whether the transaction can be run during shutdown. It can have either of these values:

`ENABLED|DISABLED`

[STORAGE_FREEZE]
Whether storage freeze is on for the transaction. It can have either of these values:

`YES|NO`

[SHUTDOWN_DISABLEOVERRIDE]
Whether to override a SHUTDOWN setting of DISABLED for the transaction definition. It can have either of these values:

YES|NO

Output parameters

[TRANDEF_TOKEN]
Token returned to represent the new transaction. definition instance.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_TRANSACTION_ID RUNAWAY_LIMIT_INVALID UNKNOWN_TCLASS
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	RUNAWAY_LIMIT_EXPECTED TCLASS_NAME_EXPECTED INVALID_FUNCTION

XMXD gate, INQUIRE_TRANDEF function

The INQUIRE_TRANDEF function of the XMXD gate is used to inquire upon a named transaction definition.

Input parameters

INQ_TRANSACTION_ID
Transaction-id to inquire upon.

TRANDEF_TOKEN
Or alternatively token representing transaction definition to inquire upon.

[USE_DTRTRAN]
If the INQ_TRANSACTION_ID cannot be found then indicates whether the DTRTRAN, if installed, should be used for the inquire instead. It can have either of these values:

YES|NO

Output parameters

[TRANSACTION_ID]
Transaction identifier.

[INITIAL_PROGRAM]
Initial program of transaction.

[PROFILE_NAME]
Profile of transaction.

[TWASIZE]
Size of Transaction Work Area.

[TRAN_PRIORITY]
Transaction priority.

[STATUS]
The status of the transaction. It can have either of these values:

ENABLED|DISABLED

[PARTITIONSET]
The partitionset defined for the transaction. It can have any of these values:

NONE|NAMED|KEEP|OWN

[PARTITIONSET_NAME]
The name of the user defined partitionset used by the transaction.

[TASKDATAKEY]
The storage key that task-lifetime storage is allocated in. It can have either of these values:

CICS|USER

[TASKDATALOC]
The location of task-lifetime storage. It can have either of these values:

BELOW|ANY

[STORAGE_CLEAR]
Whether task-lifetime storage is to be cleared before it is freemained. It can have either of these values:

YES|NO

[SYSTEM_RUNAWAY]
Whether the transaction uses the default system runaway limit. It can have either of these values:

YES|NO

[RUNAWAY_LIMIT]
The runaway limit associated with the transaction.

[DYNAMIC]
Whether the transaction is defined to be dynamic. It can have either of these values:

YES|NO

[LOCAL_QUEUING]
Whether the transaction is eligible to queue locally when it is started on the remote system. It can have either of these values:

YES|NO

[REMOTE]
Whether the transaction is remote. It can have either of these values:

YES|NO

[REMOTE_SYSTEM]
The system that a remote transaction is to be routed to.

[REMOTE_NAME]
The name of a remote transaction on the remote system.

[TRAN_ROUTING_PROFILE]
Profile to be used to route a remote transaction to a remote system.

[TCLASS]
Whether the transaction belongs to a tclass. It can have either of these values:

YES|NO

[TCLASS_NAME]
The name of the tclass that the transaction belongs to.

[INDOUBT]
The action to take if work performed by the transaction becomes indoubt. It can have any of these values:

BACKOUT|COMMIT|WAIT

[RESTART]
Whether the transaction is restartable. It can have either of these values:

YES|NO

[SPURGE]
Whether the transaction is system-purgeable. It can have either of these values:

YES|NO

[DTIMEOUT]
The deadlock timeout value for the transaction.

[TPURGE]
Whether the transaction can be purged after a terminal error. It can have either of these values:

YES|NO

[DUMP]
Whether transaction dumps are to be taken. It can have either of these values:

YES|NO

[TRACE]
The level of tracing associated with the transaction. It can have any of these values:

STANDARD|SPECIAL|SUPPRESSED

[SHUTDOWN]
Whether the transaction can be run during shutdown. It can have either of these values:

ENABLED|DISABLED

[RESSEC]
Whether resource security checking is active. It can have either of these values:

YES|NO

[CMDSEC]
Whether command security checking is active. It can have either of these values:

YES|NO

[STORAGE_FREEZE]
Whether storage freeze is on for the transaction. It can have either of these values:

YES|NO

[ISOLATE]
Whether the transaction runs in its own subspace. It can have either of these values:

YES|NO

[SYSTEM_ATTACH]
Whether a system task will be attached using this transaction definition It can have either of these values:

YES|NO

[DTRTRAN]
Indicates whether the returned transaction definition is the dynamic transaction routing transaction definition or not. It can have either of these values:

YES|NO

TCB_HISTORY
returns historical data indicating the frequency of usage of ic each subspace-inheriting open TCB mode by tasks with the caller's these transaction id.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UNKNOWN_TRANSACTION_ID
INVALID	INVALID_TOKEN INVALID_FUNCTION
DISASTER	LOGIC_ERROR ABEND LOOP

XMXD gate, INQUIRE_REMOTE_TRANDEF function

The INQUIRE_REMOTE_TRANDEF function of the XMXD gate is used to inquire upon a remote transaction definition.

Input parameters

REMOTESYSTEM_KEY
Remote system of remote transaction definition to be found.

REMOTENAME_KEY
Remote name of remote transaction definition to be found.

Output parameters

[TRANSACTION_ID]
Transaction identifier.

[INITIAL_PROGRAM]
Initial program of transaction.

[PROFILE_NAME]
Profile of transaction.

[TWASIZE]
Size of Transaction Work Area.

[TRAN_PRIORITY]
Transaction priority.

[STATUS]
The status of the transaction. It can have either of these values:

ENABLED|DISABLED

[PARTITIONSET]
The partitionset defined for the transaction. It can have any of these values:

NONE|NAMED|KEEP|OWN

[PARTITIONSET_NAME]
The name of the user defined partitionset used by the transaction.

[TASKDATAKEY]
The storage key that task-lifetime storage is allocated in. It can have either of these values:

CICS|USER

[TASKDATALOC]
The location of task-lifetime storage. It can have either of these values:

BELOW|ANY

[STORAGE_CLEAR]
Whether task-lifetime storage is to be cleared before it is freemained. It can have either of these values:

YES|NO

[SYSTEM_RUNAWAY]
Whether the transaction uses the default system runaway limit. It can have either of these values:

YES|NO

[RUNAWAY_LIMIT]
The runaway limit associated with the transaction.

[DYNAMIC]
Whether the transaction is defined to be dynamic. It can have either of these values:

YES|NO

[LOCAL_QUEUING]
Whether the transaction is eligible to queue locally when it is started on the remote system. It can have either of these values:

YES|NO

[REMOTE]
Whether the transaction is remote. It can have either of these values:

YES|NO

[REMOTE_SYSTEM]
The system that a remote transaction is to be routed to.

[REMOTE_NAME]
The name of a remote transaction on the remote system.

[TRAN_ROUTING_PROFILE]
Profile to be used to route a remote transaction to a remote system.

[TCLASS]
Whether the transaction belongs to a tclass. It can have either of these values:

YES|NO

[TCLASS_NAME]
The name of the tclass that the transaction belongs to.

[INDOUBT]
The action to take if work performed by the transaction becomes indoubt. It can have any of these values:

BACKOUT|COMMIT|WAIT

[RESTART]
Whether the transaction is restartable. It can have either of these values:

YES|NO

[SPURGE]
Whether the transaction is system-purgeable. It can have either of these values:

YES|NO

[DTIMEOUT]
The deadlock timeout value for the transaction.

[TPURGE]
Whether the transaction can be purged after a terminal error. It can have either of these values:

YES|NO

[DUMP]
Whether transaction dumps are to be taken. It can have either of these values:

YES|NO

[TRACE]
The level of tracing associated with the transaction. It can have any of these values:

STANDARD|SPECIAL|SUPPRESSED

[SHUTDOWN]
Whether the transaction can be run during shutdown. It can have either of these values:

ENABLED|DISABLED

[RESSEC]
Whether resource security checking is active. It can have either of these values:

YES|NO

[CMDSEC]
Whether command security checking is active. It can have either of these values:

YES|NO

[STORAGE_FREEZE]
Whether storage freeze is on for the transaction. It can have either of these values:

YES|NO

[ISOLATE]
Whether the transaction runs in its own subspace. It can have either of these values:

YES|NO

[SYSTEM_ATTACH]
Whether a system task will be attached using this transaction definition It can have either of these values:

YES|NO

[DTRTRAN]
Indicates whether the returned transaction definition is the dynamic transaction routing transaction definition or not. It can have either of these values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	REMOTE_NOT_FOUND
INVALID	INVALID_FUNCTION
DISASTER	LOGIC_ERROR ABEND LOOP

XMXE gate, GET_TXN_ENVIRONMENT function

The GET_TXN_ENVIRONMENT function of the XMXE gate is used to acquire a transaction environment for a task that was DS instead XM attached.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	DUPLICATE_ENVIRONMENT ATTACHED_TRANSACTION
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION

XMXE gate, FREE_TXN_ENVIRONMENT function

The FREE_TXN_ENVIRONMENT function of the XMXE gate is used to release a transaction environment for a task that was DS instead XM attached.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NO_ENVIRONMENT ATTACHED_TRANSACTION
DISASTER	ABEND LOOP
INVALID	INVALID_FUNCTION

Transaction manager domain's generic gates

Table 117 summarizes the transaction manager domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 117. Transaction manager domain's generic gates

Gate	Trace	Function	Format
XMDM	XM 0101 XM 0102	PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
XMST	XM 0C01 XM 0C02	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669
Format STST—"Statistics domain's generic format" on page 1198

Transaction manager domain's generic format

Table 118 shows the generic format owned by the transaction manager domain, and shows the function performed on the call.

Table 118. Generic format owned by the transaction manager domain

Format	Calling module	Function
XMNT	DFHXMSR DFHXMAT DFHXMTA DFHXMCL	MXT_NOTIFY MXT_CHANGE_NOTIFY
XMDN	DFHXMXD DFHXMQD DFHXMDD	TRANDEF_NOTIFY TRANDEF_DELETE_QUERY
XMPP	DFHXMIQ	FORCE_PURGE_INHIBIT_QUERY

In the descriptions of the format that follow, the "input" parameters are input not to the transaction manager domain, but to the domain being called by the transaction manager. Similarly, the "output" parameters are output by the domain that was called by the transaction manager domain, in response to the call.

Format XMNT, MXT_NOTIFY function

The MXT_NOTIFY function of XMNT format is used to notify other domains when CICS is at, or no longer at, the maximum task limit for user tasks.

Input parameters

MXTQUEUING
Indicates whether queuing for MXT has just started or just stopped. It can have either of these values:

`STARTED|STOPPED`

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

Format XMNT, MXT_CHANGE_NOTIFY function

The MXT_CHANGE_NOTIFY function of XMNT format is used to notify other domains of a change to the MXT limit. The called domains indicate whether they can cope with the new limit.

Input parameters

REQUESTED_MXT
The new limit requested for MXT.

Output parameters

ALLOCATED_MXT
Indicates the limit that the called domain can cope with when the LIMIT_TOO_HIGH exception is returned.

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	LIMIT_TOO_HIGH
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	INVALID_FUNCTION

Format XMDN, TRANDEF_NOTIFY function

The TRANDEF_NOTIFY function of the XMDN format is used to notify other domains that a transaction definition has been installed, changed, or deleted. The called domain's can then modify any transaction definition related data they are keeping for that definition.

Input parameters

EVENT
Indicates the event that has caused the notify to be sent. It can have any of the following values:

`INSTALL|CHANGE|DELETE`

TRANDEF_TOKEN
Token identifying the transaction definition instance subject to the above event.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

`OK|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	INVALID_FUNCTION

Format XMDN, TRANDEF_DELETE_QUERY function

The TRANDEF_DELETE_QUERY function of the XMDN format allows other domains to object to the deletion of the named transaction. definition.

Input parameters

TRANSACTION_ID
The transaction definition subject to the delete request.

Output parameters

INHIBIT_DELETE
Indicates whether the called domain wants to inhibit the deletion of the named transaction definition. It can either of the following values:

YES|NO

INHIBIT_REASON
Indicates the reason why the called domain wants to inhibit the deletion of the named transaction definition. It can have any of the following values:

AID_PENDING|ICE_PENDING|SIT_PARAMETER

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	LOGIC_ERROR ABEND LOOP
INVALID	INVALID_FUNCTION

Format XMPP, FORCE_PURGE_INHIBIT_QUERY function

The FORCE_PURGE_INHIBIT_QUERY function of the XMPP format allows other domains to object to the force purge request for the specified transaction.

Input parameters

TRANSACTION_TOKEN
Token identifying the transaction that is subject to the force purge request.

Output parameters

INHIBIT_PURGE
Indicates whether the called domain wants to inhibit the force purge of the transaction. It can have either of the following values:

YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION

Modules

Module	Function
DFHXMAB	XM domain abend program
DFHXMAT	Handles the following requests: ATTACH
DFHXMBD	Handles the following requests: START_BROWSE_TRANDEF GET_NEXT_TRANDEF END_BROWSE_TRANDEF
DFHXMCL	Handles the following requests: ADD_REPLACE_TCLASS ADD_TCLASS INQUIRE_TCLASS SET_TCLASS DELETE_TCLASS START_BROWSE_TCLASS GET_NEXT_TCLASS END_BROWSE_TCLASS REGISTER_TCLASS_USAGE DEREGISTER_TCLASS_USAGE LOCATE_AND_LOCK_TCLASS UNLOCK_TCLASS
DFHXMDD	Handles the following requests: DELETE_TRANDEF
DFHXMDM	Handles the following requests: PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHXMDUF	XM domain offline dump formatting routine
DFHXMER	Handles the following requests: SET_DEFERRED_MESSAGE INQUIRE_DEFERRED_MESSAGE SET_DEFERRED_ABEND INQUIRE_DEFERRED_ABEND REPORT_MESSAGE ABEND_TRANSACTION
DFHXMFD	Handles the following requests: FIND_PROFILE
DFHXMIQ	Handles the following requests: INQUIRE_TRANSACTION SET_TRANSACTION START_BROWSE_TRANSACTION GET_NEXT_TRANSACTION END_BROWSE_TRANSACTION START_BROWSE_TXN_TOKEN GET_NEXT_TXN_TOKEN END_BROWSE_TXN_TOKEN INQUIRE_TRANSACTION_TOKEN SET_TRANSACTION_TOKEN PURGE_TRANSACTION
DFHXMLD	Handles the following requests: LOCATE_AND_LOCK_TRANDEF UNLOCK_TRANDEF

Module	Function
DFHXMQC	Is an internal module which handles the following requests: TCLASS_ACQUIRE TCLASS_RELEASE TCLASS_LIMIT_CHANGE TCLASS_QUEUE_CHANGE
DFHXMQD	Is an internal module which handles the following requests: QUIESCE_TRANDEF DELETE_INSTANCE
DFHXMRP	Is an internal module which handles the following requests: DEFINITION_RECOVERY
DFHXMSR	Handles the following requests: INQUIRE_MXT SET_MXT INQUIRE_DTRTRAN SET_DTRTRAN
DFHXMST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHXMTRI	Interprets XM domain trace entries
DFHXMXD	Handles the following requests: ADD_REPLACE_TRANDEF SET_TRANDEF INQUIRE_TRANDEF INQUIRE_REMOTE_TRANDEF
DFHXMXE	Handles the following requests: GET_TXN_ENVIRONMENT FREE_TXN_ENVIRONMENT

Exits

There is one specific global user exit point in the transaction manager, XXMATT, which is called during Attach processing.

Note also that the general resource install/discard exit, XRSINDI, is also called by transaction manager to log installs and discards of transaction and tclass definitions.

For further information about both these exit points see the *CICS Customization Guide*.

Trace

The point IDs for the storage manager domain are of the form XMxxxx; the corresponding trace levels are XM 1, XM 2 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

Chapter 114. Security manager domain

The security manager domain provides an optional facility for checking user authority to run transactions and access resources.

Security manager domain's specific gates

Table 119 summarizes the security manager domain's specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 119. Security manager domain's specific gates

Gate	Trace	Function	XPI
XSAD	XS 0201	ADD_USER_WITH_PASSWORD	NO
	XS 0202	ADD_USER_WITHOUT_PASSWORD	NO
		DELETE_USER_SECURITY	NO
		INQUIRE_USER_ATTRIBUTES	NO
		VALIDATE_USERID	NO
XSFL	XS 0501	FLATTEN_USER_SECURITY	NO
	XS 0502	UNFLATTEN_USER_SECURITY	NO
		UNFLATTEN ESM_UTOKEN	NO
XSIS	XS 0301	INQUIRE_REGION_USERID	NO
	XS 0302	INQ_SECURITY_DOMAIN_PARMS	NO
		SET_SECURITY_DOMAIN_PARMS	NO
		SET_NETWORK_IDENTIFIER	NO
		SET_SPECIAL_TOKENS INQUIRE_REALM_NAME	NO
			NO
XSLU	XS 0801	GENERATE_APPC_BIND	NO
	XS 0802	GENERATE_APPC_RESPONSE	NO
		VALIDATE_APPC_RESPONSE	NO
XSPW	XS 0601	CREATE_PASSTICKET	NO
	XS 0602	INQUIRE_PASSWORD_DATA	NO
		UPDATE_PASSWORD	NO
		INQUIRE_CERTIFICATE_USERID	NO
		REGISTER_CERTIFICATE_USER	NO
XSRC	XS 0701	CHECK_CICS_RESOURCE	NO
	XS 0702	CHECK_CICS_COMMAND	NO
		CHECK_NON_CICS_RESOURCE	NO
		CHECK_SURROGATE_USER	NO
		REBUILD_RESOURCE_CLASSES	NO
XSXM	XS 0401	ADD_TRANSACTION_SECURITY	NO
	XS 0402	DEL_TRANSACTION_SECURITY	NO
		END_TRANSACTION	NO

XSAD gate, ADD_USER_WITH_PASSWORD function

The ADD_USER_WITH_PASSWORD function of the XSAD gate is used to add a user to the security domain and verify the associated password or oidcard.

Input parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the length of the USERID value.

PASSWORD_TYPE
specifies if the password is masked. It can have either of these values:

CLEAR|MASKED

[PASSWORD]
is the current password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

[PASSWORD_LENGTH]
is the 8-bit length of the PASSWORD value. This parameter is only valid if PASSWORD is also specified.

[NEW_PASSWORD]
is a new password, 1 through 10 alphanumeric characters, to be assigned to the userid (specified by the USERID value). This parameter is only valid if PASSWORD is also specified.

[NEW_PASSWORD_LENGTH]
is the 8-bit length of the NEW_PASSWORD value. This parameter is only valid if NEW_PASSWORD is also specified.

APPLID
is the application identifier for the CICS region.

[OIDCARD]
is an optional oidcard (operator identification card); a 65-byte field containing further security data from a magnetic strip reader (MSR) on 32xx devices.

[GROUPID]
is an optional identifier, 1 through 10 alphanumeric characters, of a RACF user group to which the userid (specified by the USERID value) is to be assigned.

[GROUPID_LENGTH]
is the 8-bit length of the GROUPID value. This parameter is only valid if GROUPID is also specified.

[ENTRY_PORT_NAME]
is an optional name of an entry port, 1 through 8 alphanumeric characters, to be assigned to the userid (specified by the USERID value).

[ENTRY_PORT_TYPE]
is the type of the optional entry port to be assigned to the userid (specified by the USERID value). It can have either of these values:

```
TERMINAL|CONSOLE
```

This parameter is only valid if ENTRY_PORT_NAME is also specified.

SIGNON_TYPE
is the type of signon for the userid (specified by the USERID value). It can have any of these values:

```
ATTACH_SIGN_ON|DEFAULT_SIGN_ON|IRC_SIGN_ON|
LU61_SIGN_ON|LU62_SIGN_ON|NON_TERMINAL_SIGN_ON|
PRESET_SIGN_ON|USER_SIGN_ON|XRF_SIGN_ON
```

Output parameters

SECURITY_TOKEN
is the token identifying the userid.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	APPLICATION_NOTAUTH ENTRY_PORT_NOTAUTH ESM_INACTIVE ESM_TRANQUIL GETMAIN_FAILURE GROUP_ACCESS_REVOKED INVALID_GROUPID INVALID_NEW_PASSWORD OIDCARD_NOTAUTH OIDCARD_REQUIRED PASSWORD_REQUIRED PASSWORD_EXPIRED PASSWORD_NOTAUTH SECLABEL_FAILURE SECURITY_INACTIVE UNKNOWN_ESM_ERROR USERID_NOT_IN_GROUP USERID_REVOKED USERID_NOT_DEFINED INVALID_USERID
DISASTER	ABEND LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSAD gate, ADD_USER_WITHOUT_PASSWORD function

The ADD_USER_WITHOUT_PASSWORD function of the XSAD gate is used to add a user to the security domain *without* verification of a associated password or oidcard.

Input parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the 8-bit length of the USERID value.

[GROUPID]
is an optional identifier, 1 through 10 alphanumeric characters, of a RACF user group to which the userid (specified by the USERID value) is to be assigned.

[GROUPID_LENGTH]
is the 8-bit length of the GROUPID value. This parameter is only valid if GROUPID is also specified.

[ENTRY_PORT_NAME]
is an optional name of an entry port, 1 through 8 alphanumeric characters, to be assigned to the userid (specified by the USERID value).

[ENTRY_PORT_TYPE]
is the type of the optional entry port to be assigned to the userid (specified by the USERID value). It can have either of these values:

```
TERMINAL|CONSOLE
```

This parameter is only valid if ENTRY_PORT_NAME is also specified.

SIGNON_TYPE
is the type of signon for the userid (specified by the USERID value). It can have any of these values:

```
ATTACH_SIGN_ON|DEFAULT_SIGN_ON|IRC_SIGN_ON|
LU61_SIGN_ON|LU62_SIGN_ON|NON_TERMINAL_SIGN_ON|
PRESET_SIGN_ON|USER_SIGN_ON|XRF_SIGN_ON
```

APPLID
is the application identifier for the CICS region.

Output parameters

SECURITY_TOKEN
is the token identifying the userid.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the domains response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	APPLICATION_NOTAUTH ENTRY_PORT_NOTAUTH ESM_INACTIVE ESM_TRANQUIL GETMAIN_FAILURE GROUP_ACCESS_REVOKED INVALID_GROUPID SECLABEL_FAILURE SECURITY_INACTIVE UNKNOWN_ESM_ERROR USERID_NOT_IN_GROUP USERID_REVOKED USERID_NOT_DEFINED INVALID_USERID
DISASTER	ABEND LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSAD gate, DELETE_USER_SECURITY function

The DELETE_USER_SECURITY function of the XSAD gate is used to delete the storage held to store the ACEE and ACEE pointer for the user represented by the security token.

Input parameters

SECURITY_TOKEN
is the token identifying the userid.

SIGNOFF_TYPE is the type of signoff for the userid identified by the SECURITY_TOKEN value. It can have any of these values:

```
ABNORMAL_SIGN_OFF|ATTACH_SIGN_OFF|DEFERRED_SIGN_OFF|
DELETE_SIGN_OFF|LINK_SIGN_OFF|NON_TERMINAL_SIGN_OFF|
PRESET_SIGN_OFF|UNFLATTEN_USER_SIGN_OFF|
USER_SIGN_OFF|XRF_SIGN_OFF
```

Output parameters

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the domains response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ESM_INACTIVE ESM_TRANQUIL INVALID_SECURITY_TOKEN SECURITY_INACTIVE SECURITY_TOKEN_IN_USE UNKNOWN_ESM_ERROR
DISASTER	ABEND LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSAD gate, INQUIRE_USER_ATTRIBUTES function

The INQUIRE_USER_ATTRIBUTES function of the XSAD gate is used to inquire about the attributes of the user represented by the security token.

Input parameters

SECURITY_TOKEN
is the token identifying the userid.

Output parameters

[USERID] is the identifier of the user (a userid of 1 through 10 alphanumeric characters). the userid (specified by the SECURITY_TOKEN value) is assigned.

USERID_LENGTH
is the length of the USERID value.

[CURRENT_GROUPID]
is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid (specified by the SECURITY_TOKEN value) is assigned.

[CURRENT_GROUPID_LENGTH]
is the 8-bit length of the GROUPID value.

[USERNAME] is an optional buffer into which the attributes of the user are placed.

[NATIONAL_LANGUAGE]
is a three-character code identifying the national language for the userid. It can have any of the values in Table 120 on page 1318.

[OPCLASS] is the operator class, in the range 1 through 24, for the userid.

[OPIDENT] is the operator identification code, 1 through 3 alphanumeric characters, for the userid.

[OPPRTY] is the operator priority value, in the range 0 through 255 (where 255 is the highest priority), for the userid.

[TIMEOUT] is the number of minutes, in the range 0 through 60, that must elapse since the user last used the terminal before CICS "times-out" the terminal.

Notes:
1. CICS rounds values up to the nearest multiple of 5.
2. A TIMEOUT value of 0 means that the terminal is not timed out.

[XRFSOFF] indicates whether or not you want CICS to sign off the userid following an XRF takeover. It can have either of these values:

FORCE|NOFORCE

[ACEE_PTR] is a pointer to the access control environment element, the control block that is generated by an external security manager (ESM) when the user signs on. If the user is not signed on, the address of the CICS DFLTUSER's ACEEis returned. If an ACEE does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	ESTAE_FAILURE EXTRACT_FAILURE INVALID_ACEE INVALID_ESM_PARAMETER INVALID_SECURITY_TOKEN NOTAUTH PROFILE_UNKNOWN SECURITY_INACTIVE
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP

Table 120. National language codes (three-characters)

Code	Language Name	Original Name
AFR	Afrikaans	Afrikaans
ARA	Arabic	Arabi
BEL	Byelorussian	Belaruskaja (mova)
BGR	Bulgarian	Bulgarski
CAT	Catalan	Catala
CHT	Traditional Chinese	Zhongwen
CHS	Simplified Chinese	
CSY	Czech	Cesky
DAN	Danish	Dansk
DEU	German	Deutsch
DES	Swiss German	Schweizer-Deutsch
ELL	Greek	Ellinika
ENA	Australian English	
ENG	UK English	English
ENU	US English	
ENP	English Upper Case	
ESP	Spanish	Espanol
FAR	Farsi	Persian
FIN	Finnish	Suomi
FRA	French	Francais
FRB	Belgian French	
FRC	Canadian French	
FRS	Swiss French	Suisse-francais
GAE	Irish Gaelic (Irish)	Gaeilge
HEB	Hebrew	Ivrith
HRV	Croatian	Hrvatski
HUN	Hungarian	Magyar
ISL	Icelandic	Islenska
ITA	Italian	Italiano
ITS	Swiss Italian	Italiano svizzero
JPN	Japanese	Nihongo
KOR	Korean	Choson-o; Hanguk-o
MKD	Macedonian	Makedonski
NLD	Dutch	Nederlands

Table 120. National language codes (three-characters) (continued)

Code	Language Name	Original Name
NLB	Belgian Dutch	
NOR	Norwegian - Bokmal	Norsk - Bokmal
NON	Norwegian - Nynorsk	Norsk - Nynorsk
PLK	Polish	Polski
PTG	Portuguese	Portugues
PTB	Brazilian Portuguese	
RMS	Rhaeto-Romanic	Romontsch
ROM	Romanian	Romana
RUS	Russian	Russkij
SHC	Serbo-Croatian (Cyr)	Srpsko-hrvatski
SHL	Serbo-Croatian (Lat)	
SKY	Slovakian	Slovensky
SLO	Slovenian	Slovenski
SRL	Serbian (Latin)	Srpski (Latin)
SRB	Serbian	Srpski
SQI	Albanian	Shqip
SVE	Swedish	Svenska
THA	Thai	Thai
TRK	Turkish	Turkce
UKR	Ukrainian	Ukrainska (mova)
URD	Urdu	Urdu

XSAD gate, VALIDATE_USERID function

The VALIDATE_USERID function of the XSAD gate is used to check whether the specified userid is valid. It is used especially when the userid has to be validated without the user being added to the system; usually because the userid was specified in a deferred START command, and the user does not need to be added to the system until the started task actually begins to execute.

Input parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the length of the USERID value.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	SECURITY_INACTIVE USERID_NOT_DEFINED USERID_NOT_DETERMINED
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSFL gate, FLATTEN_USER_SECURITY function

The FLATTEN_USER_SECURITY function of the XSFL gate is used to flatten the user's security state and place into the FLATTENED_SECURITY buffer provided.

Input parameters

SECURITY_TOKEN
is the token identifying the userid.

FLATTENED_SECURITY
is the buffer into which the flattened security state is placed.

Output parameters

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ESM_ABENDED ABEND LOOP
EXCEPTION	INVALID_SECURITY_TOKEN SECURITY_INACTIVE UNKNOWN_ESM_RESPONSE
INVALID	INVALID_FORMAT INVALID_FUNCTION INVALID_FLATTENED_BUFFER

XSFL gate, UNFLATTEN_USER_SECURITY function

The UNFLATTEN_USER_SECURITY function of the XSFL gate is used to unflatten the user security state data in the FLATTENED_SECURITY buffer, and add the userid to the security domain.

Input parameters

FLATTENED_SECURITY
is a buffer containing flattened security state data for a userid.

Output parameters

SECURITY_TOKEN
is the token identifying the userid.

ACEE_PTR
is a pointer to the access control environment element, the control block that is generated by an external security manager (ESM) when the user signs on.

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters). the userid (specified by the SECURITY_TOKEN value) is assigned.

USERID_LENGTH
is the length of the USERID value.

CURRENT_GROUPID
is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid is assigned.

CURRENT_GROUPID_LENGTH
is the 8-bit length of the GROUPID value.

ENTRY_PORT_NAME
is the name of an entry port, 1 through 8 alphanumeric characters, for the userid.

ENTRY_PORT_TYPE
is the type of the entry port for the userid. It can have either of these values:

TERMINAL|CONSOLE|NULL

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ESM_ABENDED ABEND LOOP
EXCEPTION	SECURITY_INACTIVE GETMAIN_FAILED USERID_NOT_DEFINED USERID_REVOKED USERID_NOT_IN_GROUP GROUP_ACCESS_REVOKED ENTRY_PORT_NOTAUTH APPLID_NOTAUTH SECLABEL_CHECK_FAILED ESM_INACTIVE ESM_TRANQUIL UNKNOWN_ESM_RESPONSE
INVALID	INVALID_FLATTENED_BUFFER INVALID_FORMAT INVALID_FUNCTION

XSFL gate, UNFLATTEN_ESM_UTOKEN function

The UNFLATTEN_ESM_UTOKEN function of the XSFL gate returns userid and groupid information associated with the external security manager's user token.

Input parameters

ESM_UTOKEN_PTR
is a pointer to a security manager user pointer.

Output parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters). the userid (specified by the SECURITY_TOKEN value) is assigned.

USERID_LENGTH
is the length of the USERID value.

CURRENT_GROUPID
is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid is assigned.

CURRENT_GROUPID_LENGTH
is the 8-bit length of the GROUPID value.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ESM_ABENDED ABEND LOOP
EXCEPTION	SECURITY_INACTIVE GETMAIN_FAILED USERID_NOT_DEFINED USERID_REVOKED USERID_NOT_IN_GROUP GROUP_ACCESS_REVOKED ENTRY_PORT_NOTAUTH APPLID_NOTAUTH SECLABEL_CHECK_FAILED ESM_INACTIVE ESM_TRANQUIL UNKNOWN_ESM_RESPONSE
INVALID	INVALID_FLATTENED_BUFFER INVALID_FORMAT INVALID_FUNCTION

XSIS gate, INQUIRE_REGION_USERID function

The INQUIRE_REGION_USERID function of the XSIS gate is used to return the userid and groupid associated with the jobstep that is currently executing this CICS region.

Input parameters

None.

Output parameters

REGION_USERID
is the user identifier of the CICS jobstep (a userid of 1 through 8 alphanumeric characters).

REGION_USERID_LENGTH
is the length of the REGION_USERID value.

[REGION_GROUPID]
is the identifier, 1 through 8 alphanumeric characters, of the current RACF user group to which the region userid is assigned.

[REGION_GROUPID_LENGTH]
is the 8-bit length of the REGION_GROUPID value.

RESPONSE is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSIS gate, INQ_SECURITY_DOMAIN_PARMS function

The INQ_SECURITY_DOMAIN_PARMS function of the XSIS gate is used to return the current values of parameters from the security state data.

Input parameters

None.

Output parameters

[APPLID]
is the generic applid of the CICS region

[CMDSEC]
indicates whether or the CICS region should obey the CMDSEC option specified on a transaction's resource definition. It can have either of these values:

YES|NO

[ESMEXITS]
indicates whether or not installation data is to be passed via the RACROUTE interface to the ESM for use in user exits written for the ESM. It can have either of these values:

YES|NO

[PREFIX]
returns the value of the prefix that is being applied to all resource names in authorization requests sent to the external security manager. It can contain 0 through 8 alphanumeric characters.

[PSBCHK]
indicates whether or not DL/I security checking is to be performed for a remote terminal initiating a transaction with transaction routing. It can have either of these values:

YES|NO

[RESSEC]
indicates whether the CICS region should obey the RESSEC option specified on a transaction's resource definition.

[SECURITY]
indicates whether or not security is active for this CICS region. It can have either of these values:

YES|NO

[XAPPC]
indicates whether or not session security checking is used when establishing APPC sessions. It can have either of these values:

YES|NO

[XCMD]
indicates whether or not EXEC CICS commands are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for EXEC CICS commands.

[XDB2]
indicates whether or not CICS performs DB2ENTRY security checking. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for DB2 entries.

[XDCT]
indicates whether or not destination control entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for destination control entries.

[XEJB]
indicates whether CICS support for enterprise bean security roles is enabled. It can have either of these values:

YES|NO

[XFCT]
indicates whether or not file control entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for file control entries.

[XJCT]
indicates whether or not journal entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for journal entries.

[XPCT]
indicates whether or not EXEC-started transactions entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for EXEC-started transactions entries.

[XPPT]
indicates whether or not program entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for program entries.

[XPSB]
indicates whether or not PSB entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for PSB entries.

[XTRAN]
indicates whether or not attached transaction entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for attached transaction entries.

[XTST]
indicates whether or not temporary storage entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for temporary storage entries.

XUSER
indicates whether or not user entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for user entries.

KEYRING
is the fully qualified name of the key ring that contains the keys and X.509 certificates used to support the secure sockets layer (SSL).

EJBROLE_PREFIX
is the prefix that is used to qualify the security role defined in an enterprise bean's deployment descriptor.

RESPONSE
is the domains response to the call. It can have any of these values:

`OK|DISASTER|INVALID`

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSIS gate, SET_SECURITY_DOMAIN_PARMS function

At CICS startup, loads information for the security domain from the system initialization table (SIT) into the security state data.

Input parameters

APPLID
is the generic applid of the CICS region

[CMDSEC]
indicates whether or the CICS region should obey the CMDSEC option specified on a transaction's resource definition. It can have either of these values:

YES|NO

ESMEXITS
indicates whether or not installation data is to be passed via the RACROUTE interface to the ESM for use in user exits written for the ESM. It can have either of these values:

YES|NO

[PREFIX]
specifies the prefix to be applied to resource name in any authorization requests send to the external security manager. It can be 1 through 8 alphanumeric characters, or the single character '*', which indicates that the CICS region userid is to be used as the prefix.

PSBCHK
indicates whether or not DL/I security checking is to be performed for a remote terminal initiating a transaction with transaction routing. It can have either of these values:

YES|NO

[RESSEC]
indicates whether the CICS region should obey the RESSEC option specified on a transaction's resource definition.

SECURITY
indicates whether or not security is active for this CICS region. It can have either of these values:

YES|NO

XAPPC
indicates whether or not session security checking is used when establishing APPC sessions. It can have either of these values:

YES|NO

[XCMD]
indicates whether or not EXEC CICS commands are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for EXEC CICS commands.

[XDB2]
indicates whether or not CICS performs DB2ENTRY security checking. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for DB2 entries.

[XDCT]
indicates whether or not destination control entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for destination control entries.

[XEJB]
indicates whether CICS support for enterprise bean security roles is enabled. It can have either of these values:

YES|NO

[XFCT]
indicates whether or not file control entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for file control entries.

[XJCT]
indicates whether or not journal entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for journal entries.

[XPCT]
indicates whether or not EXEC-started transactions entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for EXEC-started transactions entries.

[XPPT]
indicates whether or not program entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for program entries.

[XPSB]
indicates whether or not PSB entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for PSB entries.

[XTRAN]
indicates whether or not attached transaction entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for attached transaction entries.

[XTST]
indicates whether or not temporary storage entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for temporary storage entries.

XUSER
indicates whether or not user entries are checked by the ESM. It can have any of these values:

YES|name|NO

where *name* is your own resource class name for user entries.

KEYRING
is the fully qualified name of the key ring that contains the keys and X.509 certificates used to support the secure sockets layer (SSL).

EJBROLE_PREFIX
is the prefix that is used to qualify the security role defined in an enterprise bean's deployment descriptor.

Output parameters

RESPONSE
is the domains response to the call. It can have any of these values:

OK|DISASTER|INVALID

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	GETMAIN_FAILED KEYRING_NOT_FOUND KEYRING_NOT_AUTH
DISASTER	CWA_WAIT_PHASE_FAILURE INQUIRE_CWA_FAILURE ABEND LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSIS gate, SET_NETWORK_IDENTIFIER function

When CICS issues an OPEN ACB for VTAM, the CICS SVC is invoked to store the name (netid) of the local network combined with the local luname, and to RACLIST the profiles in the External Security Manager (ESM) APPCLU Class. If you have specified either of the SEC=NO or XAPPC=NO system initialization parameters, no action is performed, and the return code is set to OK.

If the RACLIST fails, and the CONDITIONAL parameter is NO, then CICS is terminated.

Input parameters

LOCAL_LUNAME
is the VTAM LU name of the local CICS region.

LOCAL_LUNAME_LENGTH
is the length of the VTAM LU name specified by LOCAL_LUNAME.

CONDITIONAL
indicates whether or not CICS can tolerate errors in XSIS calls due to the APPCLU profiles not being in storage (LU6.2 connections cannot be validated). It can have either of these values:

YES|NO

Output parameters

RESPONSE
is the domains response to the call. It can have any of these values:

OK|DISASTER|INVALID|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSIS gate, SET_SPECIAL_TOKENS function

The SET_SPECIAL_TOKENS function of the XSIS gate sets the security tokens for the default user ID and the region user ID.

Input parameters

DEFAULT_SECURITY_TOKEN
The security token for the default user ID.

REGION_SECURITY_TOKEN
The security token for the region user ID.

Output parameters

RESPONSE is the domains response to the call. It can have any of these values:

OK|DISASTER|INVALID

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSIS gate, INQUIRE_REALM_NAME function

Obtains the realm names under which the CICS system is executing; a realm is an environment in which a userid and password pairing is valid.

Input parameters

REALM_TYPE Indicates that the request is for the Basic realm name. Possible values are:

BASIC

Output parameters

REALM_NAME Returns the name of the realsm under which CICS is executing.

RESPONSE is the domains response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND LOOP

XSLU gate, GENERATE_APPC_BIND function

The GENERATE_APPC_BIND function of the XSLU gate generates a random number which is sent to the partner LU for partner verification.

Input parameters

None

Output parameters

RANDOM_STRING
A random eight-character string.

RESPONSE is the domains response to the call. It can have any of these values:

OK|INVALID

[REASON] is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SECURITY_INACTIVE BINDSECURITY_INACTIVE
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSLU gate, GENERATE_APPC_RESPONSE function

The GENERATE_APPC_RESPONSE function of the XSLU gate encrypts the string received from the LU partner, and generates a new random string for the partner to validate.

Input parameters

LOCAL_LUNAME
is the VTAM LU name of the local CICS region (sending the response).

REMOTE_LUNAME
is the VTAM LU name of the remote CICS region (that sent the bind).

TEST_STRING
is a random eight-character string receive with a bind request (RANDOM_STRING of the GENERATE_APPC_BIND function).

Output parameters

ENCRYPTED_TEST_STRING
is an eight-character string formed by encrypting the test string using shared DES (Data Encryption Standard/System) encryption keys.

RANDOM_STRING
is a random eight-character string.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP ESM_ABENDED ESTAE_FAILURE EXTRACT_FAILURE
EXCEPTION	NOTAUTH PROFILE_UNKNOWN PROFILE_LOCKED PROFILE_EXPIRED SESSION_KEY_NULL SECURITY_INACTIVE UNKNOWN_ESM_RESPONSE BIND_SECURITY_INACTIVE
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSLU gate, VALIDATE_APPC_RESPONSE function

The VALIDATE_APPC_RESPONSE function of the XSLU gate encrypts the string that was previously sent to the partner, and compares it with the encrypted string received from the partner.

Input parameters

LOCAL_LUNAME
is the VTAM LU name of the local CICS region (validating the response).

REMOTE_LUNAME
is the VTAM LU name of the remote CICS region (that sent the response).

TEST_STRING
is a random eight-character string receive with a validate request (RANDOM_STRING of the GENERATE_APPC_RESPONSE function).

ENCRYPTED_TEST_STRING
is an eight-character string formed by encrypting the test string using shared DES (Data Encryption Standard/System) encryption keys.

Output parameters

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the domains response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP ESM_ABENDED ESTAE_FAILURE EXTRACT_FAILURE
EXCEPTION	NOTAUTH VALIDATION_ERROR PROFILE_UNKNOWN PROFILE_LOCKED PROFILE_EXPIRED SESSION_KEY_NULL SECURITY_INACTIVE UNKNOWN_ESM_RESPONSE BIND_SECURITY_INACTIVE
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSPW gate, CREATE_PASSTICKET function

The CREATE_PASSTICKET function of the XSPW gate is used to create a RACF PassTicket (an alternative to a password). When created, the RACF PassTicket can be presented for userid verification ***once only***.

Input parameters

APPLID is the application identifier for the CICS region.

[TRANSACTION_NUMBER]
is an optional number that identifies a transaction from which the caller's security token is located. If not specified, the caller's security token is located from the principal security token associated with the current CICS task.

Output parameters

PASSTICKET is the 10-character passticket to be used for the CICS region specified by the APPLID value.

PASSTICKET_LENGTH
is the 8-bit length of the PASSTICKET value.

ESM_RESPONSE is the optional 32-bit ESM response code to the call.

ESM_REASON is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the domains response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP
EXCEPTION	FUNCTION_UNAVAILABLE PASSTICKET_NOT_CREATED SECURITY_INACTIVE TRANSACTION_NOT_FOUND UNKNOWN_ESM_ERROR
INVALID	INVALID_APPLID INVALID_FORMAT INVALID_FUNCTION

XSPW gate, INQUIRE_PASSWORD_DATA function

The INQUIRE_PASSWORD_DATA function of the XSPW gate provides information from the ESM.

Input parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) requesting the ESM information.

USERID_LENGTH
is the length of the USERID value.

PASSWORD
is the password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

PASSWORD_LENGTH
is the 8-bit length of the PASSWORD value.

[PASSWORD_TYPE]
indicates whether the password is masked. It can have either of these values:

CLEAR|MASKED

OPTIMIZE
indicates whether the user's revoke status is ignored. It can have any of these values:

YES|NO

Output parameters

[DAYS_LEFT]
is the number of days left before the password must be changed.

[PASSWORD_FAILURES]
is the number of times that the user has unsuccessfully entered tried to enter the password.

[EXPIRY_ABSTIME]
is the date and time of when the password will expire.

[LASTUSE_ABSTIME]
is the date and time of when the password was last used.

[CHANGE_ABSTIME]
is the date and time of when the password was last changed.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP ESM_ABENDED ESTAE_FAILURE EXTRACT_FAILURE
EXCEPTION	ESM_INACTIVE PASSWORD_NOTAUTH SECURITY_INACTIVE UNKNOWN_ESM_ERROR NOTAUTH USERID_UNDEFINED PASSWORD_EXPIRED USERID_REVOKED USERID_FORMAT_ERROR APPLID_NOT_AUTH
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSPW gate, UPDATE_PASSWORD_DATA function

The UPDATE_PASSWORD_DATA function of the XSPW gate assigns a new password to the userid, if the current password is input correctly and the new password meets ESM and installation defined password quality rules.

Input parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) requesting the ESM information.

USERID_LENGTH
is the length of the USERID value.

PASSWORD
is the current password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

PASSWORD_LENGTH
is the 8-bit length of the PASSWORD value.

NEW_PASSWORD
is the new password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

NEW_PASSWORD_LENGTH
is the 8-bit length of the NEW_PASSWORD value.

Output parameters

SAF_RESPONSE
is the optional 32-bit SAF response code to the call.

SAF_REASON
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

ESM_RESPONSE
is the optional 32-bit ESM response code to the call.

ESM_REASON
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP ESM_ABENDED ESTAE_FAILURE EXTRACT_FAILURE

RESPONSE	Possible REASON values
EXCEPTION	USERID_REVOKED USERID_UNDEFINED SECLABEL_FAILURE PASSWORD_NOTAUTH INVALID_NEW_PASSWORD ESM_INACTIVE SECURITY_INACTIVE UNKNOWN_ESM_ERROR GROUP_CONNECTION_REVOKED
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSPW gate, INQUIRE_CERTIFICATE_USERID function

The INQUIRE_CERTIFICATE_USERID function of the XSPW gate obtains the userid associated with an X.509 certificate that has been installed into the External Security Manager.

Input parameters

CERTIFICATE
an X.509 certificate

Output parameters

USERID
is the identifier of the user associated with the certificate.

USERID_LENGTH
is the length of the USERID value.

ESM_RESPONSE
is the optional 32-bit ESM response code to the call.

ESM_REASON
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP ESM_ABENDED ESTAE_FAILURE EXTRACT_FAILURE
EXCEPTION	LENGTH_ERROR GETMAIN_FAILED FREEMAIN_FAILED INVALID_CERTIFICATE UNKNOWN_CERTIFICATE UNTRUSTED_CERTIFICATE NOTAUTH SECURITY_INACTIVE ESM_INACTIVE UNKNOWN_ESM_ERROR
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSPW gate, REGISTER_CERTIFICATE_USER function

The REGISTER_CERTIFICATE_USER function of the XSPW gate associates a user with an X.509 certificate that has been installed into the External Security Manager.

Input parameters

USERID
is the identifier of the user to be associated with the certificate.

USERID_LENGTH
is the length of the USERID value.

PASSWORD
is the current password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

PASSWORD_LENGTH
is the 8-bit length of the PASSWORD value.

CERTIFICATE
the X.509 certificate that is to be registered to the specified userid.

Output parameters

ESM_RESPONSE
is the optional 32-bit ESM response code to the call.

ESM_REASON
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LOOP ESM_ABENDED ESTAE_FAILURE EXTRACT_FAILURE
EXCEPTION	GETMAIN_FAILED FREEMAIN_FAILED INVALID_CERTIFICATE UNKNOWN_CERTIFICATE UNTRUSTED_CERTIFICATE NOTAUTH SECURITY_INACTIVE ESM_INACTIVE UNKNOWN_ESM_ERROR
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSRC gate, CHECK_CICS_RESOURCE function

The CHECK_CICS_RESOURCE function of the XSRC gate performs CICS resource access checks.

Input parameters

RESOURCE
is the name of the resource, padded with blanks to eight-characters.

RESOURCE_TYPE
is the type of the resource. It can have any of these values:

DB2ENTRY|FILE|JOURNALNAME|PROGRAM|PSB|TDQUEUE|
TRANSACTION|TRANSATTACH|TSQUEUE

ACCESS
is the type of access to be made on the resource. It can have any of these values:

EXECUTE|READ|UPDATE|INQUIRE|SET|COLLECT|DEFINE|
PERFORM|CREATE|DISCARD|INSTALL|DELETE

[LOGMESSAGE]
indicates (optionally) whether access failures are logged to the CSCS transient data queue and the MVS System Management Facility (SMF). It can have either of these values:

YES|NO

[FORCE]
indicates (optionally) whether or not security checking is forced regardless of the setting of RESSEC in the Security Domain’s transaction token. It can have either of these values:

YES|NO

Output parameters

[FAILING_USERID]
is the userid that failed to access the resource.

[FAILING_USERID_LENGTH]
is the length of the userid (specified by the FAILING_USERID value).

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOTAUTH

XSRC gate, CHECK_CICS_COMMAND function

The CHECK_CICS_COMMAND function of the XSRC gate performs CICS command access checks.

Input parameters

RESOURCE_TYPE
is the type of the resource. It can have any of these values:

AUTINSTMODEL|AUTOINSTALL|CFDTPOOL|CONNECTION|
DB2CONN|DB2ENTRY|DB2TRAN|DELETSHIPPED|
DOCTEMPLATE|DSNAME|DUMP|DUMPDS|ENQMODEL|
EXITPROGRAM|FEPIRESOURCE|FILE|IRBATCH|IRC|
JOURNALNAME|JOURNALMODEL|LINE|LSRPOOL|MAPSET|
MODENAME|MONITOR|NONVTAM|PARTITIONSET|PARTNER|
PROCESSTYPE|PROFILE|PROGRAM|PSB|REQID|
REQUESTMODEL|RESETTIME|RRMS|SECURITY|SESSIONS|
SHUTDOWN|STATISTICS|STORAGE|STREAMNAME|
SYSDUMPCODE|SYSTEM|TASK|TCLASS|TCPIP|TCPIPSERVICE|
TDQUEUE|TERMINAL|TIME|TRACE|TRACEDEST|TRACEFLAG|
TRACETYPE|TRANCLASS|TRANDUMPCODE|TRANSACTION|
TRANSATTACH|TSMODEL|TSPOOL|TSQUEUE|TYPETERM|UOW|
UOWDSNFAIL|UOWENQ|UOWLINK|VOLUME|VTAM|WEB|
CORBASERVER|DJAR|JVMPOOL|EXCI|BEAN|BRFACILITY|
DISPATCHER|CLASSCACHE|JVM|JVMPOOL|JVMPROFILE

ACCESS is the type of access to be made on the resource. It can have any of these values:

COLLECT|DEFINE|DISCARD|INQUIRE|PERFORM|SET|CREATE|INSTALL|DELETE

[LOGMESSAGE] indicates (optionally) whether access failures are logged to the CSCS transient data queue and the MVS System Management Facility (SMF). It can have either of these values:

YES|NO

[FORCE] indicates (optionally) whether or not security checking is forced regardless of the setting of RESSEC in the Security Domain's transaction token. It can have either of these values:

YES|NO

Output parameters

[FAILING_USERID]
is the userid that failed to access the resource.

[FAILING_USERID_LENGTH]
is the length of the userid (specified by the FAILING_USERID value).

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOTAUTH

XSRC gate, CHECK_SURROGATE_USER function

The CHECK_SURROGATE_USER function of the XSRC gate performs surrogate user checking.

Input parameters

USERID
is the identifier of the surrogate user (a userid of 1 through 10 alphanumeric characters).

USERID_LENGTH
is the length of the USERID value.

ACCESS
is the type of access requested. It can have any of these values:

INSTALL|START|CHANGE

Output parameters

[FAILING_USERID]
is the userid that failed to access the resource.

[FAILING_USERID_LENGTH]
is the length of the userid (specified by the FAILING_USERID value).

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOTAUTH

XSRC gate, CHECK_NON_CICS_RESOURCE function

The CHECK_NON_CICS_RESOURCE function of the XSRC gate performs non-CICS resource access checks.

Input parameters

RESOURCE_NAME
is the address and length of the resource name, in the form RESOURCE_NAME(addr,length).

CLASSNAME
is the ESM class name in which the resource is defined.

ACCESS
is the type of access to be made on the resource. It can have any of these values:

ALTER|CONTROL|READ|UPDATE

[LOGMESSAGE]
indicates (optionally) whether access failures are logged to the CSCS transient data queue and the MVS System Management Facility (SMF). It can have either of these values:

YES|NO

Output parameters

[FAILING_USERID]
is the userid that failed to access the resource.

[FAILING_USERID_LENGTH]
is the length of the userid (specified by the FAILING_USERID value).

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the domains response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	NOTAUTH ESM_NOT_PRESENT ESM_INACTIVE RESOURCE_NOT_FOUND CLASS_NOT_FOUND INVALID_RESOURCE_NAME

XSRC gate, REBUILD_RESOURCE_CLASSES function

The REBUILD_RESOURCE_CLASSES function of the XSRC gate rebuilds the resource-class profiles.

Input parameters

None.

Output parameters

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the domains response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SECURITY_INACTIVE REBUILD_ERROR REBUILD_ALREADY_ACTIVE REBUILD_NOT_NEEDED ESM_INACTIVE

XSXM gate, ADD_TRANSACTION_SECURITY function

The ADD_TRANSACTION_SECURITY function of the XSXM gate sets the transaction options input to be stored as extended security tokens maintained by the transaction manager.

Input parameters

[PRINCIPAL_SECURITY_TOKEN]
is the optional principal security token.

[SESSION_SECURITY_TOKEN]
is the optional session security token.

[EDF_SECURITY_TOKEN]
is the optional EDF security token.

Output parameters

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

RESPONSE	Possible REASON values
DISASTER	GETMAIN_FAILED
EXCEPTION	NO_SECURITY_TOKEN
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSXM gate, DEL_TRANSACTION_SECURITY function

The DEL_TRANSACTION_SECURITY function of the XSXM gate deletes the security token of the specified token type for the transaction.

Input parameters

TOKEN_TYPE
is the type of security token for the transaction. It can have any of these values:

PRINCIPAL|SESSION|EDF

Output parameters

RESPONSE
is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

XSXM gate, END_TRANSACTION function

The END_TRANSACTION function of the XSXM gate deletes transaction-related data.

Input parameters

None

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION

Security manager domain's generic gates

Table 121 summarizes the security manager domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 121. Security manager domain's generic gates

Gate	Trace	Function	Format
XSDM	XS 0101 XS 0102	INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain's generic formats" on page 669

In initialization processing, the security manager domain performs internal routines, and sets the initial security options, as for "XSIS gate, SET_SECURITY_DOMAIN_PARMS function" on page 1325.

For all starts the information comes from the system initialization parameters.

Security manager domain also issues console messages during initialization to report whether or not security is active.

In quiesce and termination processing, the security manager domain performs internal routines only.

Modules

Module	Function
DFHXSAD	Handles the following requests: ADD_USER_WITH_PASSWORD ADD_USER_WITHOUT_PASSWORD DELETE_USER_SECURITY INQUIRE_USER_ATTRIBUTES VALIDATE_USERID
DFHXSDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHXSDUF	XS domain offline dump formatting routine
DFHXSFL	Handles the following requests: FLATTEN_USER_SECURITY UNFLATTEN_USER_SECURITY UNFLATTEN_ESM_UTOKEN
DFHXSIS	Handles the following requests: INQUIRE_SECURITY_DOMAIN_PARMS INQUIRE_REGION_USERID SET_SECURITY_DOMAIN_PARMS SET_NETWORK_IDENTIFIER SET_SPECIAL_TOKENS INQUIRE_REALM_NAME

Module	Function
DFHXSLU	Handles the following requests: GENERATE_APPC_BIND GENERATE_APPC_RESPONSE VALIDATE_APPC_RESPONSE
DFHXSPW	Handles the following requests: INQUIRE_PASSWORD_DATA UPDATE_PASSWORD CREATE_PASSTICKET INQUIRE_CERTIFICATE_USERID REGISTER_CERTIFICATE_USER
DFHXSRC	Handles the following requests: CHECK_CICS_RESOURCE CHECK_CICS_COMMAND CHECK_NON_CICS_RESOURCE CHECK_SURROGATE_USER REBUILD_RESOURCE_CLASSES
DFHXSSA	Manages the routing of all security domain supervisor requests, and handles those requests that are concerned with adding and deleting users.
DFHXSSB	Handles all the supervisor state interfaces with the ESM that are concerned with extracting data from the ESM's database.
DFHXSSC	Handles all the supervisor state interfaces with the ESM that are concerned with resource checking, including the building and deleting of in-storage profiles for the use of the resource check functions.
DFHXSSD	Handles supervisor state interfaces with RACF that are concerned with PassTicket generation.
DFHXSSI	Handles the following requests: DEACTIVATE_SECURITY INITIALIZE_SECURITY_SVC TERMINATE_SECURITY_SVC
DFHXSTRI	Interprets XS domain trace entries

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the security manager domain are of the form XS xxxx; the corresponding trace levels are XS 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*.

External interfaces

The following external call is used by the security manager:

- MVS RACROUTE macro to request ESM services

Part 4. CICS modules

This part contains:

Chapter 115. CICS directory

This section lists, in alphanumeric order by element name, the contents of the distribution tapes listed in Table 122.

The list shows, for each element:
- The name of the element
- The type of element
- A description of the element
- The names of the source and object distribution libraries containing the element.

Classification of elements

Name

This is the name of the element in the distribution library.

Type

The types of elements are:

CSECT.
A control section or, in the case of a source element only, the first part of a control section (other source elements may be copied by the CSECT). Where an object module is OCO, this is indicated following the type CSECT; no source code is provided for modules thus classified.

DSECT.
A dummy section (or appropriate high-level language equivalent) defining a CICS data area.

Macro.
A macro definition.

Source.
Source code that is not a CSECT.

Sample.
Sample tables, programs, map sets, partition sets, or data files.

Symbolic.
A definition (with no DSECT statement) of a CICS data area, or a group of EQU statements that symbolically define values used throughout a program.

Other.
Job control language statements or cataloged procedures. See the *CICS Transaction Server for z/OS Installation Guide* and the *CICS System Definition Guide* for the handling of these elements.

Library

Two columns are given under the heading **Library**. These correspond to source code and object code distribution respectively. The distribution tapes are in SMP/E RELFILE format, and a RELFILE number indicates the position of each data set on a particular tape. For further details about the format of the distribution tapes, see the *CICS Transaction Server for z/OS Program Directory*.

Some elements have several COBOL, PL/I, C, and assembler-language versions with the same name; these elements are shown here as cataloged in more than one source distribution library.

The meanings of the letters in the library columns is given in Table 122.

Table 122. CICS Transaction Server for z/OS, Version 3 Release 1 distribution tapes

Letter	Tape volser	File name	Library
02	CI6100	HCI6100.F2	CICSTS31.CICS.ADFHINST
03	CI6100	HCI6100.F3	CICSTS31.CICS.ADFHMOD *
04	CI6100	HCI6100.F4	CICSTS31.CICS.ADFHAPD1

Table 122. CICS Transaction Server for z/OS, Version 3 Release 1 distribution tapes (continued)

Letter	Tape volser	File name	Library
05	CI6100	HCI6100.F5	CICSTS31.CICS.ADFHAPD2
06	CI6100	HCI6100.F6	CICSTS31.CICS.ADFHCLIB
07	CI6100	HCI6100.F7	CICSTS31.CICS.ADFHCOB
08	CI6100	HCI6100.F8	CICSTS31.CICS.ADFHAC370
09	CI6100	HCI6100.F9	CICSTS31.CICS.ADFHENV
10	CI6100	HCI6100.F10	CICSTS31.CICS.ADFHLANG
11	CI6100	HCI6100.F11	CICSTS31.CICS.ADFHMAC
12	CI6100	HCI6100.F12	CICSTS31.CICS.ADFHMLIB
13	CI6100	HCI6100.F13	CICSTS31.CICS.ADFHMSGS
14	CI6100	HCI6100.F14	CICSTS31.CICS.ADFHMSRC
15	CI6100	HCI6100.F15	CICSTS31.CICS.ADFHPARM
16	CI6100	HCI6100.F16	CICSTS31.CICS.ADFHPLIB
17	CI6100	HCI6100.F17	CICSTS31.CICS.ADFHPL1
18	CI6100	HCI6100.F18	CICSTS31.CICS.ADFHPROC
19	CI6100	HCI6100.F19	CICSTS31.CICS.ADFHSAMP
20	CI6100	HCI6100.F20	CICSTS31.CICS.ADFHSDCK
C2	CI6100	JCI6101.F1	CICSTS31.CICS.ADFHCOB
C3	CI6100	JCI6101.F2	COBOL elements of CICSTS31.CICS.ADFHSAMP
C4	CI6100	JCI6101.F2	COBOL elements of CICSTS31.CICS.ADFHMOD
P2	CI6100	JCI6102.F1	CICSTS31.CICS.ADFHPLI
P3	CI6100	JCI6102.F2	PL/I elements of CICSTS31.CICS.ADFHSAMP
D2	CI6100	JCI6103.F1	CICSTS31.CICS.ADFHC370
D3	CI6100	JCI6103.F2	C elements of CICSTS31.CICS.ADFHSAMP
OS	CI610S	CICSTS31.CICS.OPTSRC01	-

An asterisk (*) following the RELFILE number indicates that the distribution library contains object modules.

Note: Object modules only are supplied for the Japanese language feature; corresponding source code is ***not*** provided for these modules.

Optional listings

Assembled listings of programs and source listings of macros, DSECTs, and symbolic definitions are available with CICS, and can be supplied on CD-ROM or microfiche. For further information about the optional listings, see the *CICS Transaction Server for z/OS Program Directory*

Contents of the distribution tapes

Table 123. CICS modules directory

Name	Type	Description	Library	
ACCTINDX	Sample	Primer - batch index file recovery - COBOL	C3	-
ACCTREC	Sample	Primer - account record - COBOL	C3	-
ACCTSET	Sample	Primer - map set - COBOL	19	-
ACCT00	Sample	Primer - menu display - COBOL	C3	-
ACCT01	Sample	Primer - initial request processing - COBOL	C3	-
ACCT02	Sample	Primer - update processing - COBOL	C3	-
ACCT03	Sample	Primer - requests for printing - COBOL	C3	-
ACCT04	Sample	Primer - error processing - COBOL	C3	-
ACIXREC	Sample	Primer - index record - COBOL	C3	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
AXMBF	CSECT	Buffer management routine	-	03
AXMER	CSECT	Server task error recovery	-	03
AXMEV	CSECT	Event control and task management routine	-	03
AXMEV1	CSECT	Event management MVS POST exit	-	03
AXMFL	CSECT	Sequential file I/O routine	-	03
AXMHP	CSECT	Heap storage routine	-	03
AXMHS	CSECT	Hash value generation subroutine	-	03
AXMLF	CSECT	Server environment LIFO storage routine	-	03
AXMLFMVS	CSECT	LIFO storage routine - MVS batch version	-	03
AXMLK	CSECT	Lock management routine	-	03
AXMMS	CSECT	Message editing and processing routine	-	03
AXMMSTAB	CSECT	Message filtering table	-	03
AXMOP	CSECT	Operator communication routine	-	03
AXMOS	CSECT	Server operating system interface	-	03
AxphG	CSECT	Page storage routine	-	03
AXMRM	CSECT	Resource manger initialization/termination	-	03
AXMRS	CSECT	Resource tracking routine	-	03
AXMSC	CSECT	Server connection routine	-	03
AXMSC1	CSECT	Locate server connection system area	-	03
AXMSC2	CSECT	Server connection services interface	-	03
AXMSI	CSECT	Subsystem initialization routine	-	03
AXMTI	CSECT	Timer interval service	-	03
AXMTK	CSECT	Task attach and detach routine	-	03
AXMTM	CSECT	Mode-independent time and date service	-	03
AXMTR	CSECT	Server trace management routine	-	03
AXMVS	CSECT	Variable sized shared storage routine	-	03
AXMWH	CSECT	AXMWH - data areas	-	03
AXMWT	CSECT	AXMWT - data areas	-	03
AXMXM	CSECT	Cross memory interface	-	03
AXMXM1	CSECT	Cross memory interface POST module	-	03
CALLDLI	Macro	CALL DL/I services	11	-
CAUBLD	CSECT	CAU builder front end	-	03
CAUBLDIN	CSECT	CAU builder input processor	-	03
CAUBLDMR	CSECT	CAU builder merge processor	-	03
CAUBLDOT	CSECT	CAU builder output processor	-	03
CAUCAFBE	CSECT	CAU CAFB abend exit	-	03
CAUCAFB1	CSECT	CAU CAFB main program	-	03
CAUCAFB2	CSECT	CAU CAFB data save program	-	03
CAUCAFDT	CSECT	CAU CAFF date utility	-	03
CAUCAFFE	CSECT	CAU CAFF abend exit	-	03
CAUCAFF1	CSECT	CAU CAFF main program	-	03
CAUCAFF2	CSECT	CAU CAFF options	-	03
CAUCAFF3	CSECT	CAU CAFF start program	-	03
CAUCAFF4	CSECT	CAU CAFF stop program	-	03
CAUCAFF5	CSECT	CAU CAFF pause program	-	03
CAUCAFF6	CSECT	CAU CAFF continue program	-	03
CAUCAFF7	CSECT	CAU CAFF help program	-	03
CAUCAFP	CSECT	CAU CAFB request handler	-	03
CAUJCLBL	Sample	Sample JCL for running CAU builder	02	-
CAUJCLCA	Sample	Sample JCL for CAU Affinity data files	02	-
CAUJCLCC	Sample	Sample JCL for CAU Affinity control file	02	-
CAUJCLLD	Sample	Sample JCL for running CAU scanner (Detail mode)	02	-
CAUJCLLS	Sample	Sample JCL for running CAU scanner (Summary mode)	02	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
CAUJCLRP	Sample	Sample JCL for running CAU Reporter	02	-
CAULMS	CSECT	CAU load module scanner	-	03
CAUMAP1	CSECT	CAU BMS map CAFF01	-	03
CAUMAP1U	CSECT	CAU BMS map CAFF01	-	19
CAUMAP2	CSECT	CAU BMS map CAFF02	-	03
CAUMAP2U	CSECT	CAU BMS map CAFF02	-	19
CAUMAP3	CSECT	CAU BMS map CAFFH1	-	03
CAUMAP4	CSECT	CAU BMS map CAFFH2	-	03
CAUMSGCS	CSECT	CAU message manager CICS stub	-	03
CAUMSGMN	CSECT	CAU message manager	-	03
CAUMSGTB	CSECT	CAU message table	-	03
CAUREP	CSECT	CAU reporter main module	-	03
CAUREPFM	CSECT	CAU reporter file manager	-	03
CAUREPPM	CSECT	CAU reporter print manager	-	03
CAUREPRM	CSECT	CAU reporter report manager	-	03
CAUTABM	CSECT	CAU detector table manager	-	03
CAUTABS	CSECT	CAU detector table storage manager	-	03
CAUXDUMM	CSECT	CAU detector dummy exit	-	03
CAUXITBA	CSECT	CAU detector BAM process exit	-	03
CAUXITBB	CSECT	CAU detector BAM activity exit	-	03
CAUXITB1	CSECT	CAU detector XBADEACT exit	-	03
CAUXITIR	CSECT	CAU detector pseudo-conv end exit	-	03
CAUXITI1	CSECT	CAU detector TRUE	-	03
CAUXITML	CSECT	CAU detector logoff exit	-	03
CAUXITMS	CSECT	CAU detector signoff exit	-	03
CAUXITM1	CSECT	CAU detector XMEOUT exit	-	03
CAUXITOA	CSECT	CAU detector ADDRESS exit	-	03
CAUXITOC	CSECT	CAU detector CANCEL exit	-	03
CAUXITOE	CSECT	CAU detector ENQ/DEQ exit	-	03
CAUXITOG	CSECT	CAU detector GETMAIN exit	-	03
CAUXITOL	CSECT	CAU detector LOAD/RELEASE exit	-	03
CAUXITOQ	CSECT	CAU detector TS exit	-	03
CAUXITOR	CSECT	CAU detector RETRIEVE exit	-	03
CAUXITOS	CSECT	CAU detector SPI exit	-	03
CAUXITOW	CSECT	CAU detector WAIT exit	-	03
CAUXITOY	CSECT	CAU detector LOAD/FREEMAIN exit	-	03
CAUXITO1	CSECT	CAU detector XEIOUT exit	-	03
CAUXITXX	CSECT	CAU detector ICE expiry exit	-	03
CAUXITX1	CSECT	CAU detector XICEXP exit	-	03
CMC	Symbolic	SAA communications pseudonyms for C	D2	-
CMCOBOL	Symbolic	SAA communications pseudonyms for COBOL	C2	-
CMHASM	Symbolic	SAA communications pseudonyms for assembler	11	-
CMPLI	Symbolic	SAA communications pseudonyms for PL/I	P2	-
DFHABAB	CSECT	AP domain abend handling	-	03
DFHABABA	DSECT	ABAB parameter list	OS	-
DFHABABM	Macro	ABAB request	OS	-
DFHABABT	CSECT	ABAB trace interpretation data	-	03
DFHABEND	Macro	Issue an ABEND macro	OS	-
DFHABREV	CSECT	String abbreviation checker	OS	03
DFHACP	CSECT	Abnormal condition program	OS	03
DFHACPTB	Macro	ACP abend table	OS	-
DFHADINS	CSECT	AD EJB CICS resource definitions	-	03
DFHADJAR	CSECT	AD JAR to DJAR mapping	-	03
DFHADSTR	CSECT	AD JAR to DJAR mapping	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHADUR@	CSECT		-	03
DFHADURM	Sample	Sample URM to set CICS user id (C version)	19	-
DFHAFCD	Macro	Authorized function control block (AFCB)	11	-
DFHAFCS	Macro	Authorized function common storage anchor	OS	-
DFHAIBD	Macro	Application interface control block	11	-
DFHAICB	Macro	Application interface control block	11	-
DFHAICBP	CSECT	Application interface control block module	OS	03
DFHAID	Symbolic	3270 attention identifiers	11	07
DFHAID	Symbolic	3270 attention identifiers - COBOL	C2	-
DFHAID	Symbolic	3270 attention identifiers - PL/I	P2	-
DFHAID	Symbolic	3270 attention identifiers - C/370	D2	-
DFHAIDDS	DSECT	Automatic initiate descriptor	11	-
DFHAIDUF	CSECT (OCO)	Autoinstall terminal model manager (AITMM) SDUMP formatter	-	03
DFHAIINA	DSECT	AIIN parameter list	OS	-
DFHAIINM	Macro	AIIN request	OS	-
DFHAIINT	CSECT (OCO)	AIIN trace interpretation data	-	03
DFHAIIN1	CSECT (OCO)	AITMM - initialization management program	-	03
DFHAIIN2	CSECT (OCO)	AITMM - initialization subtask program	-	03
DFHAIIQ	CSECT (OCO)	AITMM - locate/unlock/inquire/browse	-	03
DFHAIIQA	DSECT	AIIQ parameter list	OS	-
DFHAIIQM	Macro	AIIQ request	OS	-
DFHAIIQT	CSECT (OCO)	AIIQ trace interpretation data	-	03
DFHAIRP	CSECT (OCO)	AITMM - initialization/recovery	-	03
DFHAIRPA	DSECT	AIRP parameter list	OS	-
DFHAIRPM	Macro	AIRP request	OS	-
DFHAIRPT	CSECT (OCO)	AIRP trace interpretation data	-	03
DFHAITDS	DSECT	AITMM - static storage	OS	-
DFHAITM	CSECT (OCO)	AITMM - add replace/delete	-	03
DFHAITMA	DSECT	AITM parameter list	OS	-
DFHAITMM	Macro	AITM request	OS	-
DFHAITMT	CSECT (OCO)	AITM trace interpretation data	-	03
DFHALP	CSECT	Terminal allocation	OS	03
DFHALRC	CSECT	Automatic initiate descriptor recovery	-	03
DFHALXM	CSECT	AL XM transaction attach	-	03
DFHAM	Macro	Address mode switching macro	-	11
DFHAMBA	CSECT	RDO install of Processtype resources	-	03
DFHAMCSD	CSECT	RDO command logger	-	03
DFHAMDH	CSECT	RDO install of Document resources	-	03
DFHAMD2	CSECT		-	03
DFHAMEJ	CSECT	RDO install of EJB objects	OS	03
DFHAMER	CSECT	RDO error message builder	-	03
DFHAMFC	CSECT	RDO install for FCT resources	-	03
DFHAMGL	CSECT	RDO list generator	-	03
DFHAMLM	CSECT	Program to install log manager objects	-	03
DFHAMNQ	CSECT	RDO install of Enqmodel resources	-	03
DFHAMOP	CSECT	RDO install of Requestmodel resources	-	03
DFHAMPAB	CSECT	RDO AMP error handler	OS	03
DFHAMPAD	CSECT	RDO add command	-	03
DFHAMPAP	CSECT	RDO append command	-	03
DFHAMPCH	CSECT	RDO check command	-	03
DFHAMPCO	CSECT	RDO copy and rename commands	-	03
DFHAMPC1	CSECT	SPI generic names match	-	03
DFHAMPC2	CSECT	SPI check list name and produce list of groups	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHAMPC3	CSECT	SPI diagnose duplicate objects	-	03
DFHAMPDF	CSECT	RDO define/redefine command	OS	03
DFHAMPDI	CSECT	RDO display command	OS	03
DFHAMPDL	CSECT	RDO delete/remove commands	OS	03
DFHAMPEN	CSECT	RDO end AMP handler	OS	03
DFHAMPEX	CSECT	RDO expand command	OS	03
DFHAMPFI	CSECT	RDO begin AMP handler	OS	03
DFHAMPG	CSECT	RDO install of PG resources	-	03
DFHAMPIL	CSECT	RDO install command	OS	03
DFHAMPLO	CSECT	RDO lock/unlock command	OS	03
DFHAMPN	CSECT	RDO install for partner resources	OS	03
DFHAMPVW	CSECT	RDO view command	OS	03
DFHAMP00	CSECT	RDO allocation manager (DFHAMP)	OS	03
DFHAMRDI	CSECT	RDO install logger	OS	03
DFHAMSN	CSECT	RDO set name/type/set/stype from arg list	OS	03
DFHAMSO	CSECT	RDO install of TCPIP services	-	03
DFHAMST	CSECT	RDO update time and date in arg list	OS	03
DFHAMTD	CSECT	Program to install Transient Data objects	-	03
DFHAMTP	CSECT	RDO AMP request processor	OS	03
DFHAMTS	CSECT	RDO install of Tsmodel resources	OS	03
DFHAMXM	CSECT	Install XM domain resources (transaction and tranclass objects)	OS	03
DFHANRAT	Macro	3270 attribute character resolution	11	-
DFHANRWC	Macro	3270 control character resolution	11	-
DFHAPAC	DSECT	AP domain abnormal condition reporting interface	-	03
DFHAPACA	DSECT	APAC parameter list	OS	-
DFHAPACM	Macro	APAC request	OS	-
DFHAPACT	CSECT	APAC translate table	-	03
DFHAPAPA	DSECT	APAP parameter list	OS	-
DFHAPAPM	Macro	APAP request	OS	-
DFHAPAPT	CSECT	APAP trace interpretation data	OS	03
DFHAPATT	CSECT	AP domain - entrypoint attach	-	03
DFHAPCBT	CSECT		-	03
DFHAPDDS	DSECT	DFHAPDM static storage	OS	-
DFHAPDM	CSECT	AP domain - initialization/termination	-	03
DFHAPDN	CSECT	AP domain - transaction definition notify	-	03
DFHAPDUF	CSECT (OCO)	AP domain - formatted dump print	-	03
DFHAPEVI	Macro	AP domain - environment initialization	OS	-
DFHAPEX	CSECT	AP domain - user exit service	-	03
DFHAPEXA	DSECT	APEX parameter list	OS	-
DFHAPEXM	Macro	APEX request	OS	-
DFHAPEXT	CSECT	APEX trace interpretation data	OS	03
DFHAPH8@	CSECT		-	03
DFHAPH8O	CSECT	Java hotpooling runtime options 0 H8 PIPI	19	03
DFHAPID	DSECT	Inquire on AP data	-	03
DFHAPIDS	DSECT	Interval control static storage	OS	-
DFHAPIDT	DSECT		-	03
DFHAPIN	CSECT	AP domain - special initialization for programs and user-replaceable modules	OS	03
DFHAPIQ	CSECT (OCO)	AP domain - user exit data access service	-	03
DFHAPIQT	CSECT (OCO)	APIQ trace interpretation data	-	03
DFHAPIQX	Macro	APIQ request	11	-
DFHAPIQY	DSECT	APIQ parameter list	11	-
DFHAPJC	CSECT	AP domain - journal interface gate service	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHAPLH1	CSECT		-	03
DFHAPLH3	CSECT		-	03
DFHAPLIA	CSECT	AP domain - language interface program	OS	-
DFHAPLIT	CSECT (OCO)	AP domain - language interface service	-	03
DFHAPLI1	CSECT (OCO)	AP domain - language interface functions 1	-	03
DFHAPLI2	CSECT (OCO)	AP domain - language interface functions 2	-	03
DFHAPLI3	CSECT (OCO)	AP domain - language interface functions 3	-	03
DFHAPLI4	CSECT		-	03
DFHAPLI5	CSECT		-	03
DFHAPLI6	CSECT		-	03
DFHAPLI7	CSECT		-	03
DFHAPLJ1	CSECT		-	03
DFHAPLJ3	CSECT		-	03
DFHAPNT	CSECT	AP domain - MXT notify gate	OS	03
DFHAPPG	CSECT	AP domain - optimize initial_link for DFHMIRS	-	03
DFHAPPIS	CSECT	Java hotpooling PIPI service routines	-	03
DFHAPPIV	CSECT	Java hotpooling PIPI service routines	-	03
DFHAPRC	CSECT	User log record recovery module	-	03
DFHAPRDA	CSECT	APRD interface parameter area	OS	-
DFHAPRDR	CSECT	Resource definition recovery gate	-	03
DFHAPRDT	CSECT	APRD translate table	-	03
DFHAPRT	CSECT	AP Domain - route transaction gate	OS	03
DFHAPRTA	DSECT	APRT parameter list	OS	-
DFHAPRTM	Macro	APRT request	OS	-
DFHAPRTT	CSECT	APRM trace interpretation data	OS	03
DFHAPSDF	CSECT	AP domain - formatted dump print module	OS	03
DFHAPSI	CSECT	AP domain - gate initialization	OS	03
DFHAPSIP	CSECT	AP domain - system initialization program	OS	03
DFHAPSM	CSECT	AP domain - storage notify gate	OS	03
DFHAPST	CSECT	AP domain - statistics collection	OS	03
DFHAPTC	CSECT	AP domain - TC transport for Requeststreams	-	03
DFHAPTCA	CSECT	APTC interface parameter area	OS	-
DFHAPTCM	CSECT	APTC interface macro	OS	-
DFHAPTCT	CSECT		-	03
DFHAPTC1	CSECT	AP TC trace interpretation	-	03
DFHAPTI	CSECT	AP domain - timer notify gate	OS	03
DFHAPTIM	CSECT	AP domain - interval control midnight task	OS	03
DFHAPTIX	CSECT	AP domain - expiry analysis task	OS	03
DFHAPTPA	Symbolic	IRC trace point ID aliases	OS	-
DFHAPTRA	CSECT	IRC trace interpreter	OS	03
DFHAPTRB	CSECT	XRF trace interpreter	OS	03
DFHAPTRC	CSECT	User exit trace interpreter	OS	03
DFHAPTRD	CSECT	DFHAPDM/DFHAPAP trace interpreter	OS	03
DFHAPTRE	CSECT (OCO)	Data tables trace interpreter	-	03
DFHAPTRF	CSECT (OCO)	SAA communications and resource recovery interfaces trace interpreter	-	03
DFHAPTRG	CSECT	ZC exception and VTAM exit trace interpreter	OS	03
DFHAPTRI	CSECT	AP domain - trace interpretation router	OS	03
DFHAPTRJ	CSECT	ZC VTAM interface trace interpreter	OS	03
DFHAPTRK	CSECT	AP domain - resource definition interpretation module	-	03
DFHAPTRL	CSECT	CICS OS/2 LU2 mirror trace interpreter	OS	03
DFHAPTRN	CSECT (OCO)	Autoinstall terminal model manager trace interpreter	-	03
DFHAPTRO	CSECT	LU6.2 application request logic trace interpreter	OS	03
DFHAPTRP	CSECT	Program control trace interpreter	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHAPTRR	CSECT (OCO)	Partner resource manager trace interpreter	-	03
DFHAPTRS	CSECT (OCO)	AP domain - DFHEISR trace interpreter	-	03
DFHAPTRU	CSECT	ZC install trace interpretation	OS	03
DFHAPTRV	CSECT (OCO)	AP domain - DFHSRP trace interpreter	-	03
DFHAPTRW	CSECT (OCO)	AP domain - FEPI trace interpreter	-	03
DFHAPTRX	CSECT	ZC persistent sessions trace interpretation	OS	03
DFHAPTRY	CSECT	AP domain - trace formatting (APRM, APXM, ICXM, and TDXM)	OS	03
DFHAPTR0	CSECT	Trace interpreter for old-style AP trace	OS	03
DFHAPTR2	CSECT	AP domain - statistics trace interpreter	OS	03
DFHAPTR5	CSECT	File control trace interpreter	OS	03
DFHAPTR6	CSECT	DBCTL trace interpreter	OS	03
DFHAPTR7	CSECT	Transaction routing trace interpreter	OS	03
DFHAPTR8	CSECT	Security trace interpreter	OS	03
DFHAPTR9	CSECT	Interval control trace interpreter	OS	03
DFHAPUEA	DSECT	APUE parameter list	OS	-
DFHAPUEM	Macro	APUE request	OS	-
DFHAPUET	CSECT	APUE trace interpretation data	OS	03
DFHAPXDD	CSECT	AP domain - transaction definition extension	OS	-
DFHAPXM	CSECT	AP domain - transaction initialization and termination services	OS	03
DFHAPXMA	DSECT	APXM parameter list	OS	-
DFHAPXME	CSECT	AP domain - XM exception handler	OS	03
DFHAPXMT	CSECT (OCO)	APXM trace interpretation data	-	03
DFHASMVS	Other	Cataloged procedure to assemble CICS programs and user-written macro-level programs	18	-
DFHASSUA	DSECT	ASSU parameter list	OS	-
DFHASSUM	Macro	ASSU request	OS	-
DFHASSUT	CSECT	ASSU trace interpretation data	OS	03
DFHASV	CSECT	Authorized services interface	OS	03
DFHATUP	CSECT	Audit trail Utility Program	-	03
DFHAUDUF	CSECT		-	03
DFHAUPLE	Other	Cataloged procedure to assemble and link-edit CICS control tables, and provide information to SMP/E	02	-
DFHAUTH	Macro	Verify environment and activate CICS SVCs	OS	-
DFHAXI	Macro	XRF alternate subsystem identifier table	OS	-
DFHA03DS	DSECT	VTAM statistics	11	-
DFHA03DS	DSECT	VTAM statistics - COBOL	C2	07
DFHA03DS	DSECT	VTAM statistics - PL/I	P2	-
DFHA04DS	DSECT	Autoinstall statistics	11	-
DFHA04DS	DSECT	Autoinstall statistics - COBOL	C2	07
DFHA04DS	DSECT	Autoinstall statistics - PL/I	P2	-
DFHA06DS	DSECT	Terminal statistics	11	-
DFHA06DS	DSECT	Terminal statistics - COBOL	C2	07
DFHA06DS	DSECT	Terminal statistics - PL/I	P2	-
DFHA08DS	DSECT	LSR pool statistics	11	-
DFHA08DS	DSECT	LSR pool statistics - COBOL	C2	07
DFHA08DS	DSECT	LSR pool statistics - PL/I	P2	-
DFHA09DS	DSECT	LSR pool file-related statistics	11	-
DFHA09DS	DSECT	LSR pool file-related statistics	C2	07
DFHA09DS	DSECT	LSR pool file-related statistics	P2	-
DFHA14DS	DSECT	ISC/IRC statistics for system entries	11	-
DFHA14DS	DSECT	ISC/IRC statistics for system entries	C2	07
DFHA14DS	DSECT	ISC/IRC statistics for system entries	P2	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHA16DS	DSECT	Table manager statistics	11	-
DFHA16DS	DSECT	Table manager statistics	C2	07
DFHA16DS	DSECT	Table manager statistics	P2	-
DFHA17DS	DSECT	File control statistics	11	-
DFHA17DS	DSECT	File control statistics	C2	07
DFHA17DS	DSECT	File control statistics	P2	-
DFHA20DS	DSECT	ISC/IRC statistics for mode entries	11	-
DFHA20DS	DSECT	ISC/IRC statistics for mode entries	C2	07
DFHA20DS	DSECT	ISC/IRC statistics for mode entries	P2	-
DFHA21DS	DSECT	ISC/IRC attach-time statistics	11	-
DFHA21DS	DSECT	ISC/IRC attach-time statistics	C2	07
DFHA21DS	DSECT	ISC/IRC attach-time statistics	P2	-
DFHA22DS	DSECT	FEPI pool statistics	11	-
DFHA22DS	DSECT	FEPI pool statistics	C2	07
DFHA22DS	DSECT	FEPI pool statistics	P2	-
DFHA23DS	DSECT	FEPI connection statistics	11	-
DFHA23DS	DSECT	FEPI connection statistics	C2	07
DFHA23DS	DSECT	FEPI connection statistics	P2	-
DFHA24DS	DSECT	FEPI target statistics	11	-
DFHA24DS	DSECT	FEPI target statistics	C2	07
DFHA24DS	DSECT	FEPI target statistics	P2	-
DFHBAAC	CSECT	BAAC CDURUN and Gate module	-	03
DFHBAACT	CSECT	BAM Activity Class class declaration	-	03
DFHBAAC0	CSECT		-	03
DFHBAAC1	CSECT		-	03
DFHBAAC2	CSECT		-	03
DFHBAAC3	CSECT		-	03
DFHBAAC4	CSECT		-	03
DFHBAAC5	CSECT		-	03
DFHBAAC6	CSECT		-	03
DFHBAAR1	CSECT		-	03
DFHBAAR2	CSECT		-	03
DFHBAA10	CSECT		-	03
DFHBAA11	CSECT		-	03
DFHBAA12	CSECT		-	03
DFHBABR	CSECT	BABR CDURUN and Gata Module	-	03
DFHBABRA	CSECT	BABR interface parameter area	OS	-
DFHBABRM	Macro	BABR interface macro	OS	-
DFHBABRT	CSECT		-	03
DFHBABU1	CSECT		-	03
DFHBACR	CSECT	BACR CDURUN and Gate Module	-	03
DFHBACRT	CSECT		-	03
DFHBADM	CSECT	BA Domain Management	-	03
DFHBADUF	CSECT	BA Domain Dump Formatting	-	03
DFHBADU1	CSECT		-	03
DFHBAGDT	CSECT		-	03
DFHBALR2	CSECT		-	03
DFHBALR3	CSECT		-	03
DFHBALR4	CSECT		-	03
DFHBALR5	CSECT		-	03
DFHBALR6	CSECT		-	03
DFHBALR7	CSECT		-	03
DFHBALR8	CSECT		-	03
DFHBALR9	CSECT		-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHBAM51	CSECT	CSDUP - SPI offline messages table (51xx)	OS	03
DFHBAM52	CSECT	CSDUP - SPI offline messages table (52xx)	OS	03
DFHBAM55	CSECT	CSDUP - SPI offline messages table (55xx)	OS	03
DFHBAM56	CSECT	CSDUP - SPI offline messages table (56xx)	OS	03
DFHBAOFI	CSECT		-	03
DFHBAPR	CSECT	BAPR CDURUN and Gate Module	-	03
DFHBAPRT	CSECT		-	03
DFHBAPR0	CSECT		-	03
DFHBAPT1	CSECT		-	03
DFHBAPT2	CSECT		-	03
DFHBAPT3	CSECT		-	03
DFHBARUC	CSECT		-	03
DFHBARUD	CSECT		-	03
DFHBARUP	CSECT	CBTS Repository Utility Program	-	03
DFHBASCH	CSECT	BRDATA for CBTS Constants	-	08
DFHBASCL	CSECT	BRDATA for CBTS Constants	-	17
DFHBASCO	CSECT	BRDATA for CBTS Constants	-	07
DFHBASDD	CSECT	BRDATA for CBTS Bridge Exit	11	-
DFHBASDH	CSECT	BRDATA for CBTS Bridge Exit	-	08
DFHBASDL	CSECT	BRDATA for CBTS Bridge Exit	-	17
DFHBASDO	CSECT	BRDATA for CBTS Bridge Exit	-	19
DFHBASP	CSECT	BASP Gate Module and BA Context Class	-	03
DFHBATRI	CSECT	BAM Domain Trace Interpretation	-	03
DFHBATT	CSECT	BAM CDURUN and Gate Module	-	03
DFHBATTT	CSECT		-	03
DFHBAUE	CSECT	BAUE Gate Module	-	03
DFHBAVP1	CSECT		-	03
DFHBAXM	CSECT	BA XM Interfaces	-	03
DFHBAXMT	CSECT		-	03
DFHBEPB	CSECT	RDO batch error program	OS	03
DFHBEPC	CSECT	RDO message formatting module	OS	03
DFHBFTCA	Macro	Built-in functions TCA macro	11	-
DFHBMPIC	Macro	BMS picture analysis	11	-
DFHBMS	Macro	Basic mapping support request	11	-
DFHBMSCA	Symbolic	BMS attribute definitions	11	08
DFHBMSCA	Symbolic	BMS attribute definitions	C2	07
DFHBMSU	Macro		-	18
DFHBMSUP	Macro		-	03
DFHBMUTM	Macro	Trace BMS module generation options	OS	-
DFHBPXPA	Sample		-	02
DFHBPXP0	Sample		-	02
DFHBPXP1	Sample		-	02
DFHBRACD	Symbolic	Bridge copybook	11	-
DFHBRACH	Symbolic	Bridge copybook	D2	-
DFHBRACL	Symbolic	Bridge copybook	P2	17
DFHBRACO	Symbolic	Bridge copybook	C2	-
DFHBRARD	Symbolic	Bridge copybook	11	-
DFHBRARH	Symbolic	Bridge copybook	D2	-
DFHBRARL	Symbolic	Bridge copybook	P2	17
DFHBRARO	Symbolic	Bridge copybook	C2	-
DFHBRAT	CSECT	Design Bridge - BRAT Gate Functions	-	03
DFHBRATA	CSECT	BRAT interface parameter area	OS	-
DFHBRATM	CSECT	DFHBRAT interface macro	OS	-
DFHBRATT	CSECT		-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHBRBFB	CSECT	Bridge module	OS	-
DFHBRDCD	CSECT		OS	-
DFHBRDUF	CSECT	Bridge module	-	03
DFHBRFM	CSECT	Bridge module	-	03
DFHBRFMT	Symbolic	Trace interpretation data	-	03
DFHBRIC	CSECT	Bridge module	-	03
DFHBRIQ	CSECT	Design Bridge - BRIQ Gate Functions	-	03
DFHBRIQA	CSECT	BRIQ interface parameter area	OS	-
DFHBRIQI	CSECT		OS	-
DFHBRIQM	CSECT	DFHBRIQ interface macro	OS	-
DFHBRIQT	CSECT		-	03
DFHBRIQX	Macro	Bridge XPI macro	11	-
DFHBRIQY	Symbolic	Copybook	11	-
DFHBRMCD	Symbolic	Bridge copybook	19	-
DFHBRMCH	Symbolic	Bridge copybook	D3	-
DFHBRMCL	Symbolic	Bridge copybook	P3	-
DFHBRMCO	Symbolic	Bridge copybook	C3	-
DFHBRME	CSECT	BR Exit Program	-	03
DFHBRMF	CSECT	BR Formatter Program	-	03
DFHBRMHD	Symbolic	Bridge copybook	19	-
DFHBRMHH	Symbolic	Bridge copybook	D3	-
DFHBRMHL	Symbolic	Bridge copybook	P3	-
DFHBRMHO	Symbolic	Bridge copybook	C3	-
DFHBRMQD	Symbolic	Bridge copybook	19	-
DFHBRMQH	Symbolic	Bridge copybook	D3	-
DFHBRMQL	Symbolic	Bridge copybook	P3	-
DFHBRMQO	Symbolic	Bridge copybook	C3	-
DFHBRMS	CSECT	Bridge module	-	03
DFHBRRM	CSECT	DFHBRRM Design Bridge - Recovery Manager	-	03
DFHBRSCD	Symbolic	Bridge copybook	19	-
DFHBRSCH	Symbolic	Bridge copybook	D3	-
DFHBRSCL	Symbolic	Bridge copybook	P3	-
DFHBRSCO	Symbolic	Bridge copybook	C3	-
DFHBRSDD	Symbolic	Bridge copybook	19	-
DFHBRSDH	Symbolic	Bridge copybook	D3	-
DFHBRSDL	Symbolic	Bridge copybook	P3	-
DFHBRSDO	Symbolic	Bridge copybook	C3	-
DFHBRSP	CSECT	Bridge module	-	03
DFHBRSPA	Symbolic	Bridge copybook	OS	-
DFHBRSPM	Symbolic	Bridge copybook	OS	-
DFHBRSPT	Symbolic	Bridge copybook	-	03
DFHBRTB	CSECT	Bridge Virtual Terminal Buffer	-	03
DFHBRTC	CSECT	Bridge module	-	03
DFHBRTQ	CSECT		-	03
DFHBRTRI	Macro	Bridge module	-	03
DFHBRXM	CSECT	BR XM Principal Client	-	03
DFHBSC	Macro	Generate binary search code	11	-
DFHBSG	Macro	Switch subspace request	OS	-
DFHBSIB3	CSECT	BMS 3270 builder	OS	03
DFHBSIZ1	CSECT	Add SCS support	OS	03
DFHBSIZ3	CSECT	Add DFHZCP 3270 support	OS	03
DFHBSMIR	CSECT	Build terminal session	OS	03
DFHBSMPP	CSECT	Build pipeline pool table entry	OS	03
DFHBSM61	CSECT	Generate sessions for modegroup	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library
DFHBSM62	CSECT	Build a modegroup	OS 03
DFHBSS	CSECT	Build a connection	OS 03
DFHBSSA	CSECT	Build DFHKCP support in a system entry	OS 03
DFHBSSF	CSECT	Build stats support in a system entry	OS 03
DFHBSSS	CSECT	Build security support in a system entry	OS 03
DFHBSSZ	CSECT	Build VTAM support in a system entry	OS 03
DFHBSSZG	CSECT	Add an APPC single-session	OS 03
DFHBSSZI	CSECT	Add an indirect terminal system	OS 03
DFHBSSZL	CSECT	Add a local terminal system	OS 03
DFHBSSZM	CSECT	Introduce new system to ZCP	OS 03
DFHBSSZP	CSECT	Add an APPC parallel-session	OS 03
DFHBSSZR	CSECT	Add an MRO system	OS 03
DFHBSSZS	CSECT	Add an APPC	OS 03
DFHBSSZ6	CSECT	Add an LU6.1 connection	OS 03
DFHBST	CSECT	Common TCTTE builder	OS 03
DFHBSTB	CSECT	Add a resource for BMS	OS 03
DFHBSTBL	CSECT	Add logical device support	OS 03
DFHBSTB3	CSECT	Add partition support	OS 03
DFHBSTC	CSECT	Add install-time options support	OS 03
DFHBSTD	CSECT	Add DFHDIP support	OS 03
DFHBSTE	CSECT	Add EDF support	OS 03
DFHBSTH	CSECT	EXEC interface builder	OS 03
DFHBSTI	CSECT	Add DFHICP support	OS 03
DFHBSTM	CSECT	Add DFHMGP support	OS 03
DFHBSTO	CSECT	Spooler terminal builder	OS 03
DFHBSTP3	CSECT	Add 3270-copy support	OS 03
DFHBSTS	CSECT	Add DFHSNT support	OS 03
DFHBSTT	CSECT	Add DFHKCP support	OS 03
DFHBSTZ	CSECT	Build terminal or session resource	OS 03
DFHBSTZA	CSECT	Add DFHZCP support	OS 03
DFHBSTZB	CSECT	Add or delete bind-image	OS 03
DFHBSTZC	CSECT	Add single-session to APPC	OS 03
DFHBSTZE	CSECT	Set error message writer fields	OS 03
DFHBSTZL	CSECT	Add logical device code support	OS 03
DFHBSTZO	CSECT	Add an MVS console	OS 03
DFHBSTZP	CSECT	Pipeline terminal builder	OS 03
DFHBSTZR	CSECT	Add IRC session	OS 03
DFHBSTZS	CSECT	Add an APPC session	OS 03
DFHBSTZV	CSECT	Add VTAM and IRC information	OS 03
DFHBSTZZ	CSECT	Add non-APPC session	OS 03
DFHBSTZ1	CSECT	Add remote terminal support	OS 03
DFHBSTZ2	CSECT	Remote APPC builder	OS 03
DFHBSTZ3	CSECT	Add 3270 support	OS 03
DFHBSZZ	CSECT	Add terminal or session	OS 03
DFHBSZZS	CSECT	Add session to LU6.2 support	OS 03
DFHBSZZV	CSECT	Add VTAM terminal or session	OS 03
DFHBT	Macro	Parameter sublist translation	11 -
DFHCALLA	CSECT	CZ Direct-to-CICS	- 03
DFHCAPB	CSECT	CSDUP - command analysis program (DFHCAP)	OS 03
DFHCAPC	CSECT	RDO utility - RDL command locator	OS 03
DFHCCCC	CSECT (OCO)	GC/LC domains - functions	- 03
DFHCCCCA	DSECT	CCCC parameter list	OS -
DFHCCCCM	Macro	CCCC request	OS -
DFHCCCCT	CSECT (OCO)	CCCC trace interpretation data	- 03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHCCDM	CSECT (OCO)	GC/LC domains - initialization/termination	-	03
DFHCCDUF	CSECT (OCO)	SDUMP formatter for GC/LC domains	-	03
DFHCCNV	CSECT	Data conversion for CICS OS/2 ISC users	OS	03
DFHCCNVG	CSECT	Data conversion Gate	-	03
DFHCCNVT	CSECT		-	03
DFHCCNV2	CSECT	Convert characters in multi-byte representation	OS	03
DFHCCTRI	CSECT (OCO)	Trace interpreter for GC/LC domains	-	03
DFHCCUTL	CSECT	CICS local catalog initialization program	OS	03
DFHCDBLK	Symbolic	CONVDATA area	11	D2
DFHCDBMI	Other	CDBM group file definition JCL	-	02
DFHCDBTC	Macro	Domain call argument conversion	11	-
DFHCDC	Macro	Syntax analysis and code generation for DFHxxyyM/X domain call macros	11	-
DFHCDCON	CSECT	Formatted parameter list translator	OS	03
DFHCDEDA	DSECT	CDED parameter list	OS	-
DFHCDEDM	Macro	CDED request	OS	-
DFHCDEDT	CSECT	CDED trace interpretation data	OS	03
DFHCDKRN	CSECT	KE Java to CDURUN Interface	-	03
DFHCDMIK	Macro	Domain call inner macro - generate assignments for IN keywords	11	-
DFHCDMOK	Macro	Domain call inner macro - generate assignments for OUT keywords	11	-
DFHCDSPL	Macro	Domain call inner macro - subvalues of character list	11	-
DFHCDSUB	Macro	Domain call inner macro - subvalues of sub-parameter list	11	-
DFHCDSYN	Macro	Syntax analysis on positional operands for DFHxxyyM/X domain call macros	11	-
DFHCDTST	Macro	DFHTEST inner macro	11	-
DFHCDTYP	Macro	Determine domain call argument data type	11	-
DFHCEGN	CSECT	Goodnight transaction stub	-	03
DFHCESC	CSECT	Terminal, XRF, and enable timeout routines	-	03
DFHCESD	CSECT	CICS shutdown assist program	19	-
DFHCESDP	CSECT	CICS shutdown assist program	-	03
DFHCETRA	CSECT	Trace control transaction (CETR) - main program	OS	03
DFHCETRB	CSECT	CETR - trace component flags inquire/set	OS	03
DFHCETRC	CSECT	CETR - terminal/transaction trace control	OS	03
DFHCETRD	CSECT	CETR - common subroutines	OS	03
DFHCFCF	CSECT	CFDT Server CF Interface	-	03
DFHCFCN	CSECT	CFDT Server Client Connect/Disconnect	-	03
DFHCFDF	CSECT	CFDT AXM Server Definitions	-	03
DFHCFEN	CSECT	CFDT ENF event interface	-	03
DFHCFIF	CSECT	CFDT Server Interface Module	-	03
DFHCFIQ	CSECT	CFDT Table Inquire Routines	-	03
DFHCFLW	CSECT	CFDT Server Lock Wait Routines	-	03
DFHCFMN	CSECT	CFDT Server Main Program	-	03
DFHCFMS	CSECT	CFDT Server Messages	-	03
DFHCFOC	CSECT	CFDT Server Table Open/Close	-	03
DFHCFOP	CSECT	CFDT Server Operator Command Support	-	03
DFHCFPR	CSECT	CFDT Server Parameter Processing	-	03
DFHCFRL	CSECT	CFDT Server Pool Reload routine	-	03
DFHCFRQ	CSECT	CFDT Server Record Request Routines	-	03
DFHCFRS	CSECT	CFDT ARM Restart Support	-	03
DFHCFSP	CSECT	CFDT Server Syncpoint and Restart	-	03
DFHCFST	CSECT	CFDT Server Statistics Routines	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHCFS6D	CSECT	CFDT Statistics for list structure	11	-
DFHCFS7D	CSECT	CFDT Statistics for table accesses	11	-
DFHCFS8D	CSECT	CFDT Request statistics	11	-
DFHCFS9D	CSECT	CFDT Statistics for server storage	11	-
DFHCFUL	CSECT	CFDT Server Pool Unload Routine	-	03
DFHCFXS	CSECT	CFDT Server External Security Support	-	03
DFHCHS	CSECT	CICS mirror for CICS OS/2 and CICS/VM	OS	03
DFHCJVMA	CSECT	JVM Interface assembler routines	-	03
DFHCICS	CSECT	CICS copyright information	OS	03
DFHCLID	Macro	CICS service-level identifier	11	-
DFHCLS3	CSECT (OCO)	APPC signoff transaction program	-	03
DFHCLS4	CSECT (OCO)	APPC signon transaction program	-	03
DFHCLS5	CSECT (OCO)	Connection Quiesce Protocol	-	03
DFHCLT	Macro	Command list table	11	-
DFHCLT1$	Sample	Command list table	19	03
DFHCMAC	CSECT (OCO)	ME domain - CICS messages and codes transaction (CMAC)	-	03
DFHCMACD	Other	Source data file for CMAC transaction	13	-
DFHCMACI	Other	JCL to install the CICS messages data set	02	-
DFHCMACU	Other	JCL to update the CICS messages data set	02	-
DFHCMASM	Macro	CPI pseudonym file for assembler	11	-
DFHCMC	CSECT (OCO)	CMAC transaction map set (C/370)	-	D2
DFHCMCM	CSECT (OCO)	CMAC transaction map set	-	03
DFHCMCOB	CSECT (OCO)	CMAC transaction map set (COBOL)	-	C2
DFHCMP	CSECT	CICS monitoring compatibility interface	OS	03
DFHCMPLI	CSECT (OCO)	CMAC transaction map set (PL/1)	-	P2
DFHCNEDS	Macro	TCT console control element	11	-
DFHCNV	Macro	ISC template definition	11	-
DFHCNVCA	DSECT	DFHCNV commarea layout	OS	-
DFHCNVE	Macro	DFHCNV data conversion tables	OS	-
DFHCNVH	Macro	DFHCNV data conversion tables	OS	-
DFHCNVW$	Macro		19	03
DFHCNVXX	Macro	DFHCNV data conversion related	OS	-
DFHCNV00	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV01	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV02	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV03	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV04	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV05	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV06	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV07	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV08	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV09	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV10	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV11	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV12	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV13	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV14	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV15	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV16	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV17	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV18	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV19	CSECT	DFHCNV data conversion tables	OS	03
DFHCNV20	CSECT	DFHCNV data conversion tables	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library
DFHCNV21	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV22	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV23	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV24	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV25	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV26	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV27	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV28	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV29	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV30	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV31	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV32	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV33	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV34	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV35	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV36	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV37	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV38	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV39	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV40	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV41	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV42	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV43	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV44	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV45	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV46	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV47	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV48	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV49	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV50	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV51	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV52	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV53	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV54	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV55	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV56	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV57	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV58	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV59	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV60	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV61	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV62	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV63	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV64	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV65	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV66	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV67	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV68	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV69	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV70	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV71	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV72	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV75	CSECT	DFHCNV data conversion tables	OS 03
DFHCNV76	CSECT	DFHCNV data conversion tables	OS 03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHCNV77	CSECT	DFHCNV data conversion tables	OS	03
DFHCN06A	CSECT	DFHCNV data conversion tables	OS	-
DFHCN06B	CSECT	DFHCNV data conversion tables	OS	-
DFHCN06C	CSECT	DFHCNV data conversion tables	OS	-
DFHCN06D	CSECT	DFHCNV data conversion tables	OS	-
DFHCN06E	CSECT	DFHCNV data conversion tables	OS	-
DFHCN06F	CSECT	DFHCNV data conversion tables	OS	-
DFHCN13A	CSECT	DFHCNV data conversion tables	OS	-
DFHCN13E	CSECT	DFHCNV data conversion tables	OS	-
DFHCN28A	CSECT	DFHCNV data conversion tables	OS	-
DFHCN28E	CSECT	DFHCNV data conversion tables	OS	-
DFHCN45A	CSECT	DFHCNV data conversion tables	OS	-
DFHCN45B	CSECT	DFHCNV data conversion tables	OS	-
DFHCN45E	CSECT	DFHCNV data conversion tables	OS	-
DFHCN45F	CSECT	DFHCNV data conversion tables	OS	-
DFHCN46A	CSECT	DFHCNV data conversion tables	OS	-
DFHCN46B	CSECT	DFHCNV data conversion tables	OS	-
DFHCN46E	CSECT	DFHCNV data conversion tables	OS	-
DFHCN46F	CSECT	DFHCNV data conversion tables	OS	-
DFHCOAP	Other		OS	-
DFHCOMDS	Other	JCL to delete and recreate CICS system data sets common to all regions	02	-
DFHCOMP	Macro	Generate compare equate values	OS	-
DFHCOVER	Macro	Cover page generator	11	-
DFHCPARH	CSECT (OCO)	CPIC - CMxxxx application request handler	-	03
DFHCPCAC	CSECT (OCO)	CPIC - Accept_Conversation	-	03
DFHCPCAL	CSECT (OCO)	CPIC - Allocate	-	03
DFHCPCBA	CSECT (OCO)	CPIC - Create_CPC (Accept)	-	03
DFHCPCBB	CSECT (OCO)	CPIC - Increment_Last_Convid	-	03
DFHCPCBD	CSECT (OCO)	CPIC - Delete_Conversation	-	03
DFHCPCBE	CSECT (OCO)	CPIC - Extract_Syncpoint_rc	-	03
DFHCPCBG	CSECT (OCO)	CPIC - Initialize_CPC	-	03
DFHCPCBI	CSECT (OCO)	CPIC - Create_CPC (Initialize)	-	03
DFHCPCBL	CSECT (OCO)	CPIC - Locate_CPC	-	03
DFHCPCBS	CSECT (OCO)	CPIC - Set_CPC_Log_Data	-	03
DFHCPCBT	CSECT (OCO)	CPIC - Load module branch table	-	03
DFHCPCCA	DSECT	CPCC parameter list	OS	-
DFHCPCCD	CSECT (OCO)	CPIC - Confirmed	-	03
DFHCPCCF	CSECT (OCO)	CPIC - Confirm	-	03
DFHCPCCM	Macro	CPCC request	OS	-
DFHCPCCT	CSECT (OCO)	CPCC trace interpretation data	-	03
DFHCPCDE	CSECT (OCO)	CPIC - Deallocate	-	03
DFHCPCEA	CSECT (OCO)	CPIC - Extract_Conversation_Type	-	03
DFHCPCEB	CSECT (OCO)	CPIC - Extract_Mode_Name	-	03
DFHCPCEC	CSECT (OCO)	CPIC - Extract_Partner_LU_Name	-	03
DFHCPCED	CSECT (OCO)	CPIC - Extract_Sync_Level	-	03
DFHCPCEE	CSECT (OCO)	CPIC - Extract_Conversation_State	-	03
DFHCPCFL	CSECT (OCO)	CPIC - Flush	-	03
DFHCPCFS	CSECT (OCO)	CPIC - finite state machine	-	03
DFHCPCIC	CSECT (OCO)	CPIC - Initialize_Conversation	-	03
DFHCPCLC	CSECT (OCO)	CPIC - interface to DFHLUC	-	03
DFHCPCLM	CSECT (OCO)	CPIC - build send list	-	03
DFHCPCLR	CSECT (OCO)	DFHLUC to CPIC return code conversion	-	03
DFHCPCND	CSECT (OCO)	CPIC - Send_Data	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHCPCNE	CSECT (OCO)	CPIC - Send_Error	-	03
DFHCPCN1	CSECT (OCO)	CPIC - Send_and_Buffer	-	03
DFHCPCN2	CSECT (OCO)	CPIC - Send_and_Flush	-	03
DFHCPCN3	CSECT (OCO)	CPIC - Send_and_Prep_To_Receive	-	03
DFHCPCN4	CSECT (OCO)	CPIC - Send_and_Confirm	-	03
DFHCPCN5	CSECT (OCO)	CPIC - Send_and_Deallocate	-	03
DFHCPCOJ	CSECT (OCO)	CPIC - Output_Journaling	-	03
DFHCPCPR	CSECT (OCO)	CPIC - Prepare_To_Receive	-	03
DFHCPCRA	CSECT (OCO)	CPIC - Receive mapped data	-	03
DFHCPCRB	CSECT (OCO)	CPIC - Receive GDS header	-	03
DFHCPCRC	CSECT (OCO)	CPIC - Receive basic data	-	03
DFHCPCRI	CSECT (OCO)	CPIC - Receive_Immediate	-	03
DFHCPCRS	CSECT (OCO)	CPIC - Request_To_Send	-	03
DFHCPCRV	CSECT (OCO)	CPIC - Receive	-	03
DFHCPCRW	CSECT (OCO)	CPIC - Receive_and_Wait	-	03
DFHCPCSA	CSECT (OCO)	CPIC - Set_Conversation_Type	-	03
DFHCPCSB	CSECT (OCO)	CPIC - Set_Deallocate_Type	-	03
DFHCPCSC	CSECT (OCO)	CPIC - Set_Error_Direction	-	03
DFHCPCSD	CSECT (OCO)	CPIC - Set_Fill	-	03
DFHCPCSE	CSECT (OCO)	CPIC - Set_Log_Data	-	03
DFHCPCSF	CSECT (OCO)	CPIC - Set_Mode_Name	-	03
DFHCPCSG	CSECT (OCO)	CPIC - Set_Partner_LU_Name	-	03
DFHCPCSH	CSECT (OCO)	CPIC - Set_Prepare_To_Receive	-	03
DFHCPCSI	CSECT (OCO)	CPIC - Set_Receive_Type	-	03
DFHCPCSJ	CSECT (OCO)	CPIC - Set_Return_Control	-	03
DFHCPCSK	CSECT (OCO)	CPIC - Set_Send_Type	-	03
DFHCPCSL	CSECT (OCO)	CPIC - Set_Sync_Level	-	03
DFHCPCSM	CSECT (OCO)	CPIC - Set_TP_Name	-	03
DFHCPCTE	CSECT (OCO)	CPIC - Test_Request_To_Send_Received	-	03
DFHCPDUF	CSECT (OCO)	SDUMP formatter for CP keyword	-	03
DFHCPI	CSECT (OCO)	Common programming interface (CPI) program	-	03
DFHCPINA	DSECT	CPIN parameter list	OS	-
DFHCPINM	Macro	CPIN request	OS	-
DFHCPINT	CSECT (OCO)	CPIN trace interpretation data	-	03
DFHCPIN1	CSECT (OCO)	CPI initialization management program	-	03
DFHCPIN2	CSECT (OCO)	CPI initialization subtask program	-	03
DFHCPIR	CSECT (OCO)	SRRxxxx application request processor	-	03
DFHCPLC	CSECT (OCO)	Link-edit stub for application programs using SAA communications interface	-	03
DFHCPLRR	CSECT (OCO)	Link-edit stub for application programs using SAA resource recovery interface	-	03
DFHCPOST	Macro	POST macro for extended ECBs	OS	-
DFHCPSDS	DSECT	CPI static storage	OS	-
DFHCPSPA	DSECT	CPSP parameter list	OS	-
DFHCPSPM	Macro	CPSP request	OS	-
DFHCPSPT	CSECT (OCO)	CPSP trace interpretation data	-	03
DFHCPSRH	CSECT (OCO)	CPIC - syncpoint request handler	-	03
DFHCPY	CSECT	3270 hard copy support	OS	03
DFHCRBDS	DSECT	CICS region control block	OS	-
DFHCRBU	CSECT	UOW back-to-front processor module	-	03
DFHCRC	CSECT	Interregion abnormal exit module	OS	03
DFHCRD	DSECT	Communications recovery services declares	11	-
DFHCRERI	DSECT	AP domain - Communications recovery management - resync	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHCRERP	DSECT	Perform unshunt invoked by RM	-	03
DFHCRERS	DSECT	Session failure during syncpoint	-	03
DFHCRESI	DSECT	AP domain - communication recovery management	OS	-
DFHCRIU	CSECT	IRC RMC syncpoint event processor	-	03
DFHCRL	CSECT	RMC logging back-to-front processor	-	03
DFHCRLB	CSECT	RMC bind time logging for old MRO/LU6.2	-	03
DFHCRLBA	CSECT	CRLB parameter list	OS	-
DFHCRLBM	Macro	CRLB parameter list	OS	-
DFHCRLBT	CSECT	CRLB translate tables	-	03
DFHCRNP	CSECT	Interregion connection manager	OS	03
DFHCRQ	CSECT	ATI purge program	OS	03
DFHCRR	CSECT	Interregion session recovery program	OS	03
DFHCRRSY	CSECT	Communications resynchronization	-	03
DFHCRS	CSECT	Remote scheduler program	OS	03
DFHCRSP	CSECT	CICS IRC startup module	OS	03
DFHCRT	CSECT	Transaction routing relay program for	OS	03
DFHCRTRI	CSECT	Offline trace formatting - interpretation routine parameter list	-	03
DFHCR1U	CSECT	IRC LU61 syncpoint event processor	-	03
DFHCR2U	CSECT	IRC LU62 RMC syncpoint event processor	-	03
DFHCSA	CSECT	Common system area	OS	03
DFHCSAD	Macro	Common system area	11	-
DFHCSADS	DSECT	Common system area definition	11	-
DFHCSCDS	Symbolic	CICS SVC startup return codes	OS	-
DFHCSDUF	CSECT (OCO)	SDUMP formatter for CSA and CSA optional features list	-	03
DFHCSVC	CSECT	CICS SVC startup	OS	03
DFHCTRH	CSECT	CETR transaction help screens map set	OS	03
DFHCTRM	CSECT	CETR transaction main screens map set	OS	03
DFHCTRMU	Sample		-	19
DFHCUADD	CSECT	CSDUP - add command	OS	03
DFHCUALG	CSECT	RDO off-line generic alter utility program	-	03
DFHCUALT	CSECT	CSDUP - alter command	OS	03
DFHCUAPP	CSECT	CSDUP - append command	OS	03
DFHCUCAB	CSECT	CSDUP - command analyzer (DFHCUCA)	OS	03
DFHCUCAC	CSECT	CSD manager - return and reason codes	OS	03
DFHCUCB	CSECT	CSDUP - command builder	OS	03
DFHCUCCB	CSECT	CSDUP - RDL command locator (DFHCUCC)	OS	03
DFHCUCDB	CSECT	CSDUP - default values (DFHCUCD)	OS	03
DFHCUCDC	CSECT	CSD manager - return and reason codes	OS	03
DFHCUCOG	CSECT	CSDUP - generic copy command	OS	03
DFHCUCOM	CSECT		-	03
DFHCUCOP	CSECT	CSDUP - copy command	OS	03
DFHCUCP	CSECT	CSDUP - command processor	OS	03
DFHCUCS	CSECT	CSDUP - CSD open and close	OS	03
DFHCUCSE	CSECT	CSDUP - CSD error check routine	OS	03
DFHCUCV	CSECT	CSDUP - command validation	OS	03
DFHCUDEF	CSECT	CSDUP - define command	OS	03
DFHCUERA	CSECT	CSDUP - delete/erase command	OS	03
DFHCUFA	CSECT	Offline utilities - free automatic storage	OS	03
DFHCUFAM	Macro	Offline DFHPROC - free automatic storage	OS	-
DFHCUGA	CSECT	Offline utilities - get automatic storage	OS	03
DFHCUGAM	Macro	Offline DFHPROC - get automatic storage	OS	-
DFHCUINI	CSECT	CSDUP - initialize command	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHCULIS	CSECT	CSDUP - extract and list commands	OS	03
DFHCULOC	CSECT	CSDUP - lock/unlock routine	OS	03
DFHCUMD2	CSECT		-	03
DFHCUMIG	CSECT	CSDUP - migrate command	OS	03
DFHCUMT	CSECT	CSDUP - TCT migration	OS	03
DFHCUMTD	CSECT	RDO migration utility program for the DCT	-	03
DFHCUMTS	CSECT	RDO migration utility program for the TST	OS	03
DFHCUMWR	CSECT	CSDUP - CSD record write routine	OS	03
DFHCUMXI	CSECT	SPI offline utility for handling cross reference of IBM groups	OS	03
DFHCUPRC	CSECT	RDO off line utility	OS	03
DFHCUPRO	CSECT	CSDUP - CSD upgrade routine	OS	03
DFHCURDD	CSECT	CSD utilities - delete all existing CICS- supplied groups from previous releases	OS	03
DFHCURDI	CSECT	CSD utilities - RDL for basic initialize	OS	03
DFHCURDM	CSECT	CSD utilities - RDL for maintenance	OS	03
DFHCURDS	CSECT	CSD utilities - RDL for sample definitions	OS	03
DFHCURDX	CSECT	CSD utilities - RDL for compatibility gp	OS	03
DFHCUREM	CSECT	CSDUP - remove command	OS	03
DFHCURUG	CSECT	CSDUP - upgrade command	OS	03
DFHCUSER	CSECT	CSDUP - service command	OS	03
DFHCUSHL	CSECT	CSDUP - short lock/unlock routine	OS	03
DFHCUS1	CSECT	CSD utilities - sample service request	OS	03
DFHCUUSR	CSECT		OS	03
DFHCUVER	CSECT	CSDUP - verify command	OS	03
DFHCUXRT	CSECT	RDO offline utility for building cross reference table of IBM groups	OS	03
DFHCVDAA	Symbolic	System programming command cvda names	OS	-
DFHCVTRI	CSECT	CCNV Gate trace interpretation	-	03
DFHCZTRI	CSECT	CICS Foundation Classes trace interpretation	-	03
DFHCZTRT	CSECT	Foundation classes trace interprete tables	-	03
DFHCWTO	CSECT	Write to console operator program	OS	03
DFHCXCU	CSECT	XRF catch-up transaction	OS	03
DFHC3TRI	CSECT (OCO)	Trace interpreter for DFHCLS3 trace points	-	03
DFHC5TRI	CSECT		-	03
DFHDATE	Macro	Date formatting	OS	-
DFHDBAT	CSECT	CICS-DBCTL adapter/transformer	OS	03
DFHDBCON	CSECT	CICS-DBCTL connection program	OS	03
DFHDBCR	CSECT	CICS-DBCTL XRF tracking program	OS	03
DFHDBCT	CSECT	CICS-DBCTL control program	OS	03
DFHDBCTX	CSECT	CICS-DBCTL control exit	OS	03
DFHDBDE	CSECT	CICS-DBCTL operator transaction map set	-	03
DFHDBDI	CSECT	CICS-DBCTL disable program	OS	03
DFHDBDSC	CSECT	CICS-DBCTL disconnection program	OS	03
DFHDBDUF	CSECT (OCO)	SDUMP formatter for DBCTL, local DL/I, and remote DL/I	-	03
DFHDBIE	CSECT	CICS-DBCTL inquiry screens map set	OS	03
DFHDBIK	CSECT (OCO)	CICS-DBCTL inquiry screens map set	-	03
DFHDBIQ	CSECT	CICS-DBCTL inquiry program	OS	03
DFHDBME	CSECT	CICS-DBCTL menu program	OS	03
DFHDBMOX	CSECT	CICS-DBCTL monitoring exit	OS	03
DFHDBMP	CSECT	EDF browse map set	-	03
DFHDBMS	CSECT	EDF browse map set	OS	03
DFHDBNE	CSECT	CICS-DBCTL menu screens map set	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHDBNK	CSECT (OCO)	CICS-DBCTL menu screens map set	-	03
DFHDBP	CSECT	Dynamic backout program	OS	03
DFHDBREX	CSECT	CICS-DBCTL resume exit	OS	03
DFHDBSPX	CSECT	CICS-DBCTL suspend exit	OS	03
DFHDBSSX	CSECT	CICS-DBCTL status exit	OS	03
DFHDBSTX	CSECT	CICS-DBCTL statistics exit	OS	03
DFHDBTI	CSECT	EXEC DLI LD table	OS	03
DFHDBTOX	CSECT	CICS-DBCTL token exit	OS	03
DFHDBUCA	DSECT	COMMAREA passed to DFHDBUEX	11	-
DFHDBUDS	DSECT	DBCTL unsolicited statistics	11	07
DFHDBUDS	DSECT	DBCTL unsolicited statistics	C2	-
DFHDBUEX	CSECT	User-replaceable CICS-DBCTL exit	19	03
DFHDC	Macro	Dump service request	11	-
DFHDCPR	CSECT	Transaction dump macro-compatibility program	OS	03
DFHDCRDS	DSECT	Transaction dump control record format	OS	-
DFHDCT	Macro	Destination control table	11	-
DFHDCTD	Macro	Destination control table	11	-
DFHDCTDS	DSECT	Destination control table	11	-
DFHDDBR	CSECT (OCO)	DD domain - browse Services	-	03
DFHDDBRT	CSECT (OCO)	DDBR trace interpretation data	-	03
DFHDDDI	CSECT (OCO)	DD domain - directory services	-	03
DFHDDDIA	CSECT (OCO)	DDDI parameter list	OS	-
DFHDDDIM	CSECT (OCO)	DDDI parameter list	OS	-
DFHDDDIT	CSECT (OCO)	DDDI trace interpretation data	-	03
DFHDDDM	CSECT (OCO)	DD domain - domain services	-	03
DFHDDDU	CSECT (OCO)	DD domain - dump browse services	-	03
DFHDDDUF	CSECT (OCO)	DD domain - dump formatting	-	03
DFHDDLO	CSECT (OCO)	DD domain - locate service	-	03
DFHDDLOA	CSECT (OCO)	DDLO parameter list	OS	-
DFHDDLOM	CSECT (OCO)	DDLO parameter list	OS	-
DFHDDLOT	CSECT (OCO)	DDLO trace interpretation data	-	03
DFHDDTRI	CSECT (OCO)	DD domain - trace interpretation	-	03
DFHDEFDS	Other	JCL to delete and recreate CICS system data sets unique to each region	02	-
DFHDEIST	CSECT	DEIS trace interpretation data	-	03
DFHDESVT	DSECT	DESV trace interpretation data	-	03
DFHDFST	CSECT		OS	03
DFHDHDH	CSECT	Document Handler Domain	-	03
DFHDHDHT	CSECT		-	03
DFHDHDM	CSECT	Document Handler Domain	-	03
DFHDHDUF	CSECT	DH Document System Dump Formatter	-	03
DFHDHEI	CSECT	DH Document Template EXEC resources	-	03
DFHDHPB	CSECT		-	03
DFHDHPD	CSECT		-	03
DFHDHPM	CSECT		-	03
DFHDHPR	CSECT	DH Document Handler Read PDS routine	-	03
DFHDHPS	CSECT		-	03
DFHDHPT	CSECT		-	03
DFHDHPU	CSECT		-	03
DFHDHPX	CSECT		-	03
DFHDHRM	CSECT	DHRM CDURUN and Gate module	-	03
DFHDHRP	CSECT	Document Handler Recovery Program	-	03
DFHDHRPT	CSECT		-	03
DFHDHSL	CSECT	Document Handler Domain	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHDHSLT	CSECT		-	03
DFHDHTM	CSECT	DH Document Handler Template Manager	-	03
DFHDHTMT	CSECT		-	03
DFHDHTRI	CSECT	DH Domain Trace Formatter	-	03
DFHDHTXD	CSECT		11	-
DFHDHTXH	CSECT		-	08
DFHDHTXL	CSECT		-	17
DFHDHTXO	CSECT		-	07
DFHDHUE	CSECT	Document Domain (DH) user Exit Services	-	03
DFHDI	Macro	Data interchange request	11	-
DFHDIBDS	Macro	Data interchange	OS	-
DFHDIP	CSECT	Data interchange program	OS	03
DFHDIPDY	CSECT	Data interchange program (dummy)	OS	03
DFHDITOP	Macro	Data interchange internal macro	OS	-
DFHDKMRA	DSECT	DKMR parameter list	OS	-
DFHDKMRM	Macro	DKMR request	OS	-
DFHDKMRT	CSECT	DKMR trace interpretation data	-	03
DFHDKTRI	CSECT (OCO)	DD domain - trace interpreter	-	03
DFHDLI	CSECT	DL/I call router	OS	03
DFHDLIAI	CSECT	Application interface for DL/I	OS	03
DFHDLIDP	CSECT	DBCTL call processor	OS	03
DFHDLIRP	CSECT	DL/I remote call processor	OS	03
DFHDLLO@	CSECT		-	03
DFHDLP	Macro	CICS-DL/I interface	11	-
DFHDLPSB	Macro	Generate DL/I PSB directory list	11	-
DFHDLXDF	CSECT	DU domain - transaction dump formatter for DL/I related areas	OS	03
DFHDMDM	CSECT (OCO)	DM domain - domain initialization/quiesce	-	03
DFHDMDMA	DSECT	DMDM parameter list	OS	-
DFHDMDMM	Macro	DMDM request	OS	-
DFHDMDMT	CSECT (OCO)	DMDM trace interpretation data	-	03
DFHDMDS	CSECT (OCO)	DM domain - task reply handler	-	03
DFHDMDUF	CSECT (OCO)	SDUMP formatter for DM domain	-	03
DFHDMEN	CSECT (OCO)	Domain manager ENF support	-	03
DFHDMENF	CSECT (OCO)	Domain manager event notification routine	-	03
DFHDMENS	CSECT (OCO)	CICS ENF SRBEXIT	-	03
DFHDMENT	CSECT (OCO)	DMEN translation tables	-	03
DFHDMIQ	CSECT (OCO)	DM domain - browse and inquiry	-	03
DFHDMIQA	DSECT	DMIQ parameter list	OS	-
DFHDMIQM	Macro	DMIQ request	OS	-
DFHDMIQT	CSECT (OCO)	DMIQ trace interpretation data	-	03
DFHDMPB	CSECT	CSDUP - definition file (CSD) manager, batch environment router (DFHDMP batch)	OS	03
DFHDMPBA	CSECT	CSDUP - batch environment adapter	OS	03
DFHDMPC	CSECT	CSD manager - CICS environment router (DFHDMP CICS)	OS	03
DFHDMPCA	CSECT	CSD manager - CICS environment adapter	OS	03
DFHDMPH	Symbolic	DM domain - phase definitions	OS	-
DFHDMRM	CSECT (OCO)	CSD manager - CSD close routine	-	03
DFHDMSVC	CSECT (OCO)	DM domain - SVC processing routine	-	03
DFHDMTRI	CSECT (OCO)	DM domain - trace interpreter	-	03
DFHDMWQ	CSECT (OCO)	DM domain - wait queue subroutine	-	03
DFHDMWQA	DSECT	DMWQ parameter list	OS	-
DFHDMWQM	Macro	DMWQ request	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHDMWQT	CSECT (OCO)	DMWQ trace interpretation data	-	03
DFHDM01B	CSECT	CSDUP - connect (DFHDM01 batch)	OS	03
DFHDM01C	CSECT	CSD manager - connect (DFHDM01 CICS)	OS	03
DFHDM02B	CSECT	CSDUP - disconnect (DFHDM02 batch)	OS	03
DFHDM02C	CSECT	CSD manager - disconnect (DFHDM02 CICS)	OS	03
DFHDM03B	CSECT	CSDUP - write (DFHDM03 batch)	OS	03
DFHDM03C	CSECT	CSD manager - write (DFHDM03 CICS)	OS	03
DFHDM04B	CSECT	CSDUP - read (DFHDM04 batch)	OS	03
DFHDM04C	CSECT	CSD manager - read (DFHDM04 CICS)	OS	03
DFHDM05B	CSECT	CSDUP - delete (DFHDM05 batch)	OS	03
DFHDM05C	CSECT	CSD manager - delete (DFHDM05 CICS)	OS	03
DFHDM06B	CSECT	CSDUP - lock/unlock (DFHDM06 batch)	OS	03
DFHDM06C	CSECT	CSD manager - lock/unlock (DFHDM06 CICS)	OS	03
DFHDM08B	CSECT	CSDUP - setbrowse (DFHDM08 batch)	OS	03
DFHDM08C	CSECT	CSD manager - setbrowse (DFHDM08 CICS)	OS	03
DFHDM09B	CSECT	CSDUP - getnext (DFHDM09 batch)	OS	03
DFHDM09C	CSECT	CSD manager - getnext (DFHDM09 CICS)	OS	03
DFHDM10B	CSECT	CSDUP - endbrowse (DFHDM10 batch)	OS	03
DFHDM10C	CSECT	CSD manager - endbrowse (DFHDM10 CICS)	OS	03
DFHDM11B	CSECT	CSDUP - createset (DFHDM11 batch)	OS	03
DFHDM11C	CSECT	CSD manager - createset (DFHDM11 CICS)	OS	03
DFHDM12B	CSECT	CSDUP - eraseset (DFHDM12 batch only)	OS	03
DFHDM13B	CSECT	CSDUP - queryset (DFHDM13 batch)	OS	03
DFHDM13C	CSECT	CSD manager - queryset (DFHDM13 CICS)	OS	03
DFHDM15B	CSECT	CSDUP - read/write control records (DFHDM15 batch)	OS	03
DFHDM15C	CSECT	CSD manager - read/write control records (DFHDM15 CICS)	OS	03
DFHDM16B	CSECT	CSDUP - buildkey (DFHDM16 batch)	OS	03
DFHDM16C	CSECT	CSD manager - buildkey (DFHDM16 CICS)	OS	03
DFHDM17B	CSECT	CSDUP - relsekwa (DFHDM17 batch)	OS	03
DFHDM17C	CSECT	CSD manager - relsekwa (DFHDM17 CICS)	OS	03
DFHDM18B	CSECT	CSDUP - tokenize utilities (DFHDM18 batch)	OS	03
DFHDM18C	CSECT	CSD manager - tokenize utilities (DFHDM18 CICS)	OS	03
DFHDM19B	CSECT	CSDUP - free generic tokens chain (DFHDM19 batch)	OS	03
DFHDM19C	CSECT	CSD manager - free generic tokens chain (DFHDM19 CICS)	OS	03
DFHDM21B	CSECT	CSDUP - generic qualification (DFHDM21 batch)	OS	03
DFHDM21C	CSECT	CSD manager - generic qualification (DFHDM21 CICS)	OS	03
DFHDM22B	CSECT	CSDUP - resequence utility (DFHDM22 batch)	OS	03
DFHDM22C	CSECT	CSD manager - resequence utility (DFHDM22 CICS)	OS	03
DFHDM23B	CSECT	CSDUP - verify key work area (DFHDM23 batch)	OS	03
DFHDM23C	CSECT	CSD manager - verify key work area (DFHDM23 CICS)	OS	03
DFHDNSRT	Macro	Internal index sorting macro	OS	-
DFHDRX	Macro	DL/I resource table	OS	-
DFHDSAT	CSECT (OCO)	DS domain - attach, change mode, change/set priority, cancel task	-	03
DFHDSATA	DSECT	DSAT parameter list	OS	-
DFHDSATM	Macro	DSAT request	OS	-
DFHDSATT	CSECT (OCO)	DSAT trace interpretation data	-	03
DFHDSATX	Macro	DSAT request (XPI)	11	-
DFHDSATY	DSECT	DSAT parameter list (XPI)	11	-
DFHDSAUT	CSECT (OCO)	DS domain - authorized services	-	03
DFHDSB	CSECT	BMS data stream build	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHDSBA$	CSECT	BMS data stream build (standard)	OS	03
DFHDSBR	CSECT (OCO)	DS domain - browse, inquire task	-	03
DFHDSBRA	DSECT	DSBR parameter list	OS	-
DFHDSBRM	Macro	DSBR request	OS	-
DFHDSBRT	CSECT (OCO)	DSBR trace interpretation data	-	03
DFHDSB1$	CSECT	BMS data stream build (full)	OS	03
DFHDSCPX	CSECT (OCO)	POST routine for DS WAIT_MVS requests	-	03
DFHDSCSA	CSECT (OCO)	DS domain - update CSA on task dispatch	-	03
DFHDSDM	CSECT (OCO)	DS domain - initialization/termination	-	03
DFHDSDSA	DSECT	DSDS parameter list	OS	-
DFHDSDSM	Macro	DSDS request	OS	-
DFHDSDST	CSECT (OCO)	DSDS trace interpretation data	-	03
DFHDSDS2	CSECT (OCO)	DS domain - broadcast new max task limit	-	03
DFHDSDS3	CSECT (OCO)	DS domain - main dispatch loop	-	03
DFHDSDS4	CSECT (OCO)	DS domain - task purge routine	-	03
DFHDSDUF	CSECT (OCO)	SDUMP formatter for DS domain	-	03
DFHDSGDS	DSECT	DS domain - global statistics	11	07
DFHDSGDS	DSECT	DS domain - global statistics	C2	-
DFHDSIT	CSECT (OCO)	DS domain - set/inquire DS parameters	-	03
DFHDSITA	DSECT	DSIT parameter list	OS	-
DFHDSITM	Macro	DSIT request	OS	-
DFHDSITT	CSECT (OCO)	DSIT trace interpretation data	-	03
DFHDSKE	CSECT (OCO)	DS domain - kernel interfaces	-	03
DFHDSND	Macro	File control data set name	11	-
DFHDSPEX	CSECT (OCO)	DS domain - MVS POST exit stub	-	03
DFHDSRP	Sample	Distributed Dynamic Routing Program (COBOL)	-	07
DFHDSRP	Sample	Distributed Dynamic Routing Program (C)	C2	08
DFHDSRP	Sample	Distributed Dynamic Routing Program (Asm)	19	03
DFHDSSM	CSECT (OCO)	DS domain - storage notify handler	-	03
DFHDSSR	CSECT (OCO)	DS domain - suspend/resume/wait	-	03
DFHDSSRA	DSECT	DSSR parameter list	OS	-
DFHDSSRM	Macro	DSSR request	OS	-
DFHDSSRT	CSECT (OCO)	DSSR trace interpretation data	-	03
DFHDSSRV	Macro	DS domain - inline dispatcher services	OS	-
DFHDSSRX	Macro	DSSR request (XPI)	11	-
DFHDSSRY	DSECT	DSSR parameter list (XPI)	11	-
DFHDSST	CSECT (OCO)	DS domain - statistics collection	-	03
DFHDSSTX	CSECT (OCO)	DS domain - STIMERM exit	-	03
DFHDSTA	Macro	DBCTL statistics area (DFSDSTA)	OS	-
DFHDSTCB	CSECT (OCO)	DS domain - KEDS TCB_REPLY handler	-	03
DFHDSTI	CSECT	DS domain Timer Domain Gate Service Module	-	03
DFHDSTIQ	Macro	DS domain - obtain domain index of task issuing trace put	OS	-
DFHDSTRI	CSECT (OCO)	DS domain - Trace interpreter	-	03
DFHDSTSD	DSECT	DS domain - Task Area	OS	-
DFHDSUE	CSECT (OCO)	DS domain - enable/disable user exits	-	03
DFHDTCF	CSECT (OCO)	Shared data tables connect file PC function	-	03
DFHDTCP	CSECT (OCO)	Shared data tables cell pool management	-	03
DFHDTCV	CSECT (OCO)	Shared data tables connection validation	-	03
DFHDTDA	CSECT (OCO)	Shared data tables data space and ALET code	-	03
DFHDTDM	CSECT (OCO)	Shared data tables data management	-	03
DFHDTINS	CSECT (OCO)	Shared data tables initialization	-	03
DFHDTIX	CSECT (OCO)	Shared data tables index management	-	03
DFHDTLA	CSECT (OCO)	Shared data table load attach	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHDTLI	CSECT (OCO)	Shared data tables local initialization	-	03
DFHDTLX	CSECT (OCO)	Shared data tables load transaction	-	03
DFHDTPDS	DSECT	Data tables - services interface block	OS	-
DFHDTPC	CSECT (OCO)	Shared data tables program call stub	-	03
DFHDTRC	CSECT (OCO)	Shared data tables remote file connection and disconnection	-	03
DFHDTRE	CSECT (OCO)	Shared data tables remote file connection	-	03
DFHDTRI	CSECT (OCO)	Shared data tables remote environment initialization	-	03
DFHDTRM	CSECT (OCO)	Shared data tables record management	-	03
DFHDTRR	CSECT (OCO)	Shared data tables remote retrieval	-	03
DFHDTSR	CSECT (OCO)	Shared data tables shared retrieval	-	03
DFHDTSS	CSECT (OCO)	Shared data table server status	-	03
DFHDTST	CSECT (OCO)	Shared data table state services	-	03
DFHDTSVS	CSECT (OCO)	Shared data tables SVC services	-	03
DFHDTUP	CSECT (OCO)	Shared data tables update and syncpoint services	-	03
DFHDTXS	CSECT (OCO)	Shared data tables connection security	-	03
DFHDUDDA	DSECT	DUDD parameter list	OS	-
DFHDUDDM	Macro	DUDD request	OS	-
DFHDUDDT	CSECT	DUDD trace interpretation data	OS	03
DFHDUDM	CSECT	DU domain - initialization/termination	OS	03
DFHDUDT	CSECT	DU domain - dump table services	OS	03
DFHDUDTA	DSECT	DUDT parameter list	OS	-
DFHDUDTM	Macro	DUDT request	OS	-
DFHDUDTT	CSECT	DUDT trace interpretation data	OS	03
DFHDUDU	CSECT	DU domain - take system/transaction dump	OS	03
DFHDUDUA	DSECT	DUDU parameter list	OS	-
DFHDUDUF	CSECT (OCO)	SDUMP formatter for DU domain	-	03
DFHDUDUM	Macro	DUDU request	OS	-
DFHDUDUT	CSECT	DUDU trace interpretation data	OS	03
DFHDUDUX	Macro	DUDU request (XPI)	11	-
DFHDUDUY	DSECT	DUDU parameter list (XPI)	11	-
DFHDUF	CSECT (OCO)	SDUMP formatting router	-	03
DFHDUFFT	CSECT (OCO)	PRDUMP formatter - service functions	OS	03
DFHDUFT	CSECT (OCO)	Dump domain services	OS	03
DFHDUFTA	DSECT	DUFT parameter list	OS	-
DFHDUFTD	DSECT	Dump formatting routines parameter declares	OS	-
DFHDUFTM	Macro	DUFT macro	OS	-
DFHDUFTT	DSECT (OCO)	DUFT translate tables	OS	03
DFHDUFTX	Macro	DUFT macro	11	-
DFHDUFTY	DSECT	DUFT call structured parameter list	11	-
DFHDUFUT	CSECT (OCO)	SDUMP formatting - service functions	-	03
DFHDUIO	CSECT	DU domain - open/close/switch/write	OS	03
DFHDUIOA	DSECT	DUIO parameter list	OS	-
DFHDUIOM	Macro	DUIO request	OS	-
DFHDUIOT	CSECT	DUIO trace interpretation data	OS	03
DFHDUMPX	CSECT	DU domain - SDUMPX IEASDUMP.QUERY exit	OS	03
DFHDUPH	CSECT	Dump utility program - dump index summary	OS	03
DFHDUPM	CSECT	Dump utility program - module index	OS	03
DFHDUPMC	DSECT	Dump utility program - parameter block for module index routine	OS	-
DFHDUPP	CSECT	Dump utility program - I/O routines	OS	03
DFHDUPPC	DSECT	Dump utility program - parameter block for print routine	OS	-
DFHDUPR	CSECT	Dump utility program - main component	OS	03
DFHDUPS	CSECT	Dump utility program - dump selection	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHDUPSC	DSECT	Dump utility program - parameter block for dump selection routine	OS	-
DFHDUSR	CSECT	DU domain - dump services	OS	03
DFHDUSRA	DSECT	DUSR parameter list	OS	-
DFHDUSRM	Macro	DUSR request	OS	-
DFHDUSRT	CSECT	DUSR trace interpretation data	OS	03
DFHDUSU	CSECT	DU domain - subroutines	OS	03
DFHDUSUA	DSECT	DUSU parameter list	OS	-
DFHDUSUM	Macro	DUSU request	OS	-
DFHDUSUT	CSECT	DUSU trace interpretation data	OS	03
DFHDUSVC	CSECT	DU domain - SVC processing routine	OS	03
DFHDUTM	CSECT	DU domain - dump table manager	OS	03
DFHDUTRI	CSECT	Trace interpreter for DU domain	OS	03
DFHDUXD	CSECT	DU domain - transaction dump control	OS	03
DFHDUXFA	DSECT	DUXF parameter list	OS	-
DFHDUXFM	Macro	DUXF request	OS	-
DFHDUXFT	CSECT	DUXF trace interpretation data	OS	03
DFHDUXW	CSECT	DU domain - transaction dump buffer control	OS	03
DFHDUXWA	DSECT	DUXW parameter list	OS	-
DFHDUXWM	Macro	DUXW request	OS	-
DFHDUXWT	CSECT	DUXW trace interpretation data	OS	03
DFHDWE	Macro	Deferred work element	OS	-
DFHDWEDS	DSECT	Deferred work element	11	-
DFHDXACH	CSECT	CICS-DBCTL XRF subtask router	OS	03
DFHDXAX	CSECT	CICS-DBCTL XRF connection handling	OS	03
DFHDXCU	CSECT	CICS-DBCTL XRF catch-up transaction	OS	03
DFHDXSTM	CSECT	CICS-DBCTL XRF subtask manager	OS	03
DFHDXUEP	DSECT	CICS-DBCTL XRF plist to global user exits	11	-
DFHDYP	Sample	Dynamic routing program	C2	07
DFHDYP	Sample	Dynamic routing program	D2	-
DFHDYP	CSECT	User-replaceable dynamic routing program	19	03
DFHDYPDS	DSECT	COMMAREA passed to DFHDYP	11	-
DFHDYPDS	DSECT	COMMAREA passed to DFHDYP	C2	07
DFHDYPDS	DSECT	COMMAREA passed to DFHDYP	D2	-
DFHD2CC	CSECT	DB2 module	-	03
DFHD2CCT	CSECT	DB2 module	-	03
DFHD2CMP	CSECT	DB2 module	-	03
DFHD2CM0	CSECT	DB2 module	-	03
DFHD2CM1	CSECT	DB2 module	-	03
DFHD2CM2	CSECT	DB2 module	-	03
DFHD2CM3	CSECT	DB2 module	-	03
DFHD2COT	CSECT	DB2 module	-	03
DFHD2DUF	CSECT	DB2 module	-	03
DFHD2D2T	CSECT	DB2 module	-	03
DFHD2EDF	CSECT	DB2 module	-	03
DFHD2EXS	CSECT	DB2 module	-	03
DFHD2EX1	CSECT	DB2 module	-	03
DFHD2EX2	CSECT	DB2 module	-	03
DFHD2EX3	CSECT	DB2 module	-	03
DFHD2GDS	CSECT	DB2 module	11	07
DFHD2INI	CSECT	DB2 module	-	03
DFHD2IN1	CSECT	DB2 module	-	03
DFHD2IN2	CSECT	DB2 module	-	03
DFHD2LI	CSECT	CICS-DB2 stub (Language interface module)	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHD2MSB	CSECT	DB2 module	-	03
DFHD2RDS	CSECT	DB2 module	11	07
DFHD2RP	CSECT	DB2 module	-	03
DFHD2SSD	CSECT	DB2 module	OS	-
DFHD2ST	CSECT	DB2 module	-	03
DFHD2STP	CSECT	DB2 module	-	03
DFHD2STR	CSECT	DB2 module	-	03
DFHD2TM	CSECT	DB2 module	-	03
DFHD2TMT	CSECT	DB2 module	-	03
DFHD2TRI	CSECT	DB2 module	-	03
DFHEAI	CSECT	EXEC interface link-edit stub for EXEC calls in assembler language programs	OS	03
DFHEAI0	CSECT	EXEC interface link-edit stub for prolog and epilog calls in assembler language programs	OS	03
DFHEAMAA	CSECT	Assembler-language translator - advanced	OS	03
DFHEAMEE	CSECT	Assembler-language translator - error editor	OS	03
DFHEAMPA	CSECT	Assembler-language translator - primary code generation functions	OS	03
DFHEAMSA	CSECT	Assembler-language translator - source scanner	OS	03
DFHEAM02	CSECT	Assembler-language translator - initialization	OS	03
DFHEAM07	CSECT	Assembler-language translator - options card	OS	03
DFHEAM08	CSECT	Assembler-language translator - check options	OS	03
DFHEAM11	CSECT	Assembler-language translator - atomization	OS	03
DFHEBBND	Sample	Part of the CICS EJB sample	-	19
DFHEBCBJ	Sample	Part of the CICS EJB sample	-	19
DFHEBCB1	Sample	COBOL source for V2ACTDB program	-	19
DFHEBCB2	Sample	COBOL source for V2CSTDB program	-	19
DFHEBCNV	Sample	EJB Sample COMMAREA Conversion Table	-	19
DFHEBDAT	Sample	Part of the CICS EJB sample	-	19
DFHEBDEF	Sample	CICS EJB Sample Resource Definitions	-	19
DFHEBF	CSECT	EXEC interface for BIF DEEDIT command	OS	03
DFHEBGRT	Sample	Part of the CICS EJB sample	-	19
DFHEBRCT	CSECT	CBRC LD table	OS	03
DFHEBREB	Sample	Part of the CICS EJB sample	-	19
DFHEBTAB	Sample	Part of the CICS EJB sample	-	19
DFHEBTAL	Other	Cataloged procedure to translate, assemble and link-edit assembler-language application programs that use EXEC DLI and will run in a batch or CICS shared database region	-	18
DFHEBTPL	Other	Cataloged procedure to translate, compile and link-edit PL/I application programs that use EXEC DLI and will run in a batch or CICS shared database region	-	18
DFHEBTVL	Other	Cataloged procedure to translate, compile and link-edit VS COBOL II application programs that use EXEC DLI and will run in a batch or CICS shared database region	-	18
DFHEBU	CSECT	EXEC FMH construction	OS	03
DFHECADS	DSECT	Event control area for interval control elements	OS	-
DFHECALL	Macro	EXEC interface call macro for assembler-language	11	-
DFHECB	Macro	CICS posting and testing of operating system ECBs	OS	-
DFHECBAM	CSECT		OS	03
DFHECI	CSECT	EXEC interface stub for EXEC calls (COBOL)	OS	03
DFHECMAC	CSECT	COBOL translator - advanced code generation functions	OS	03
DFHECMEE	CSECT	COBOL translator - error editor	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHECMPC	CSECT	COBOL translator - primary code generation functions	OS	03
DFHECMSC	CSECT	COBOL translator - input scanner	OS	03
DFHECM02	CSECT	COBOL translator - initialization	OS	03
DFHECM07	CSECT	COBOL translator - options card	OS	03
DFHECM08	CSECT	COBOL translator - check options	OS	03
DFHECM10	CSECT	COBOL translator - analyze program	OS	03
DFHECM11	CSECT	COBOL translator - atomization	OS	03
DFHECM14	CSECT	COBOL translator - read input	OS	03
DFHECM17	CSECT	COBOL translator - generate output	OS	03
DFHEDC	CSECT	EXEC interface for dump control	OS	03
DFHEDCP	CSECT (OCO)	EXEC interface for dump system/transaction	-	03
DFHEDFBR	CSECT	Temporary-storage browse transaction, CEBR	OS	03
DFHEDFCB	CSECT	Build one page	OS	03
DFHEDFCC	CSECT	Parameter copy program	OS	03
DFHEDFCE	CSECT	Extract from one page	OS	03
DFHEDFCR	CSECT	LD table utilities	OS	03
DFHEDFCS	CSECT	CICS special cases	OS	03
DFHEDFCX	CSECT	Display unformatted arguments	OS	03
DFHEDFD	CSECT	EDF display program	OS	03
DFHEDFDL	CSECT	DL/I special cases	OS	03
DFHEDFDS	DSECT	EDF communication area	OS	-
DFHEDFE	CSECT	EDF attach error handler	OS	03
DFHEDFM	CSECT	EDF map set	OS	03
DFHEDFP	CSECT	EDF control program	OS	03
DFHEDFR	CSECT	EDF response table	OS	03
DFHEDFS	CSECT	EDF display handling routines	OS	03
DFHEDFU	CSECT	Data utilities	OS	03
DFHEDFW	CSECT	Display working storage	OS	03
DFHEDFX	CSECT	EDF task switch program	OS	03
DFHEDI	CSECT	EXEC interface for data interchange	OS	03
DFHEDMAD	CSECT	C/370 translator - advanced code generation functions	OS	03
DFHEDMEE	CSECT	C/370 translator - error editor	OS	03
DFHEDMPD	CSECT	C/370 translator - primary code generation functions	OS	03
DFHEDMSD	CSECT	C/370 translator - input scanner	OS	03
DFHEDM02	CSECT	C/370 translator - initialization	OS	03
DFHEDM07	CSECT	C/370 translator - options card	OS	03
DFHEDM08	CSECT	C/370 translator - check options	OS	03
DFHEDM10	CSECT	C/370 translator - analyze program	OS	03
DFHEDM11	CSECT	C/370 translator - atomization	OS	03
DFHEDM14	CSECT	C/370 translator - read input	OS	03
DFHEDM17	CSECT	C/370 translator - generate output	OS	03
DFHEDP	CSECT	EXEC DLI command stub	OS	03
DFHEEI	CSECT	EXEC interface for HANDLE, ADDRESS, ASSIGN	OS	03
DFHEEX	CSECT	EXEC FMH extraction	OS	03
DFHEGL	CSECT	EXEC interface for unmapped LU6.2 commands	OS	03
DFHEIACQ	CSECT (OCO)	EXEC ACQUIRE TERMINAL	-	03
DFHEIAR	Macro	EIP arguments macro	OS	-
DFHEIBAM	CSECT		-	03
DFHEIBLC	DSECT	EXEC interface block	C2	07
DFHEIBLK	DSECT	EXEC interface block	11	-
DFHEIBLK	DSECT	EXEC interface block	C2	07
DFHEICDS	DSECT	EXEC interface COMMAREA	11	-
DFHEICRE	DSECT	EXEC CICS CREATE command	-	03
DFHEIDDS	Macro	EXEC interface argument 0 descriptor	11	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHEIDH	CSECT	Document Language Table and EI Layer	-	03
DFHEIDI	CSECT	Address set for COBOL	OS	-
DFHEIDTI	CSECT	EXEC ask-time, format-time program	OS	03
DFHEIEIA	DSECT	EIEI parameter list	OS	-
DFHEIEIM	Macro	EIEI request	OS	-
DFHEIEIT	CSECT	EIEI trace interpretation data	OS	03
DFHEIEM	CSECT	DFHEIEM Design Exec EM request handler	-	03
DFHEIEND	Macro	EXEC interface storage end macro	11	-
DFHEIENT	Macro	EXEC interface prolog macro	11	-
DFHEIFC	Macro	File control exec interface module	-	03
DFHEIFSP	Macro	Free space	OS	-
DFHEIGBL	Macro	EXEC interface globals definition macro	11	-
DFHEIGDS	CSECT	Translator table (GDS commands)	OS	03
DFHEIGSP	Macro	Get space	OS	-
DFHEIIC	CSECT (OCO)	EXEC interface IC module	-	03
DFHEIIF	Macro	EXEC interface IF macro	OS	-
DFHEILIA	Other	Used by DFHEITAL cataloged procedure	11	-
DFHEILIC	Other	Used by DFHEITCL cataloged procedure	C2	-
DFHEILID	Other	Used by DFHEITDL cataloged procedure	D2	-
DFHEILIP	Other	Used by DFHEITPL cataloged procedure	P2	-
DFHEIMDS	Macro	Master terminal return codes	OS	-
DFHEIMOP	CSECT	Translator options	OS	03
DFHEIMSG	Macro	EXEC interface message macro	11	-
DFHEIMV	Macro	EXEC interface move macro	OS	-
DFHEIN00	CSECT	Interpreter - CECI/CECS program	OS	03
DFHEIN01	CSECT	Interpreter - control module	OS	03
DFHEIN02	CSECT	Interpreter - initialization	OS	03
DFHEIN03	CSECT	CBRC/CECI/CEDA/CEMT - storage manager	OS	03
DFHEIN11	CSECT	CBRC/CECI - atomization	OS	03
DFHEIN12	CSECT	Interpreter - argument analysis	OS	03
DFHEIN13	CSECT	CECI/CEDA/CEMT - diagnosis	OS	03
DFHEIN16	CSECT	CECI/CEDA/CEMT - binary conversion	OS	03
DFHEIN19	CSECT	Interpreter - command analysis	OS	03
DFHEIN20	CSECT	Interpreter - table analysis	OS	03
DFHEIN21	CSECT	Interpreter - keyword analysis	OS	03
DFHEIN22	CSECT	Interpreter - special case code	OS	03
DFHEIN23	CSECT	Interpreter - plist generation	OS	03
DFHEIN26	CSECT	CECI/CEMT - message editor	OS	03
DFHEIN27	CSECT	Interpreter - spelling correction	OS	03
DFHEIN28	CSECT	Interpreter - basic messages	OS	03
DFHEIN50	CSECT	Interpreter - special displays	OS	03
DFHEIN51	CSECT	Interpreter - display extraction	OS	03
DFHEIN52	CSECT	Interpreter - syntax display	OS	03
DFHEIN53	CSECT	Interpreter - utilities	OS	03
DFHEIN54	CSECT	Interpreter - further utilities	OS	03
DFHEIP	CSECT	EXEC (command-level) interface program	-	03
DFHEIPA	CSECT	EXEC interface prolog and epilog code for assembler-language programs	OS	03
DFHEIPAD	Macro	EXEC interface intermodule addressing	OS	-
DFHEIPDS	DSECT	EXEC interface control blocks	11	-
DFHEIPEL	Source	EXEC interface layer epilog code	OS	-
DFHEIPEQ	Symbolic	EXEC interface EQU statements	OS	-
DFHEIPER	Source	EXEC interface error handling data	OS	-
DFHEIPLR	Macro	EXEC interface epilog code	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHEIPLS	Macro	EXEC interface prolog code	OS	-
DFHEIPPL	Source	EXEC interface layer prolog code	OS	-
DFHEIPRT	CSECT (OCO)	EXEC interface for perform resettime	-	03
DFHEIPSE	CSECT (OCO)	EXEC interface for perform security	-	03
DFHEIPSH	CSECT (OCO)	EXEC interface for perform shutdown	-	03
DFHEIQBA	CSECT (OCO)	EXEC inquire reqid	-	03
DFHEIQCF	CSECT (OCO)	EXEC inquire cfdtpool	-	03
DFHEIQDH	CSECT (OCO)	EXEC inquire doctemplate	-	03
DFHEIQDN	CSECT (OCO)	EXEC inquire/set for external data sets	-	03
DFHEIQDS	CSECT (OCO)	EXEC inquire/set/discard for files	-	03
DFHEIQDU	CSECT (OCO)	EXEC inquire/set for dump data sets and dump codes	-	03
DFHEIQD2	CSECT (OCO)		-	03
DFHEIQEJ	CSECT (OCO)		-	03
DFHEIQIR	CSECT (OCO)	EXEC inquire/set for IRC	-	03
DFHEIQMS	CSECT (OCO)	EXEC inquire/set for monitor and stats	-	03
DFHEIQMT	CSECT	EXEC inquire/set for CEMT-only commands	-	03
DFHEIQOP	CSECT	EXEC inquire requestmodel	-	03
DFHEIQPF	CSECT (OCO)	EXEC inquire/discard for profiles	-	03
DFHEIQPN	CSECT (OCO)	EXEC inquire/discard for partners	-	03
DFHEIQRQ	CSECT (OCO)	EXEC inquire for queued requests (REQIDs)	-	03
DFHEIQRR	CSECT (OCO)	SPI Inquire RRMS Processor	-	03
DFHEIQSA	CSECT (OCO)	EXEC inquire/set for system attributes	-	03
DFHEIQSC	CSECT (OCO)	EXEC inquire/set for connections	-	03
DFHEIQSJ	CSECT (OCO)	EXEC inquire/set for journals or discard for journalnames	-	03
DFHEIQSK	CSECT (OCO)	EXEC inquire/set for tasks	-	03
DFHEIQSL	CSECT (OCO)	EXEC inquire/for journalmodel or streamname or discard for journalmodel	-	03
DFHEIQSM	CSECT (OCO)	EXEC inquire/set for modenames	-	03
DFHEIQSO	CSECT (OCO)	EXEC inquire tcpip	-	03
DFHEIQSP	CSECT (OCO)	EXEC inquire/set/discard for programs	-	03
DFHEIQSQ	CSECT (OCO)	EXEC inquire/set for TD queues	-	03
DFHEIQST	CSECT (OCO)	EXEC inquire/set for terminals	-	03
DFHEIQSV	CSECT (OCO)	EXEC inquire/set for volumes	-	03
DFHEIQSX	CSECT (OCO)	EXEC inquire/set/discard for transactions	-	03
DFHEIQSY	CSECT (OCO)		-	03
DFHEIQSZ	CSECT (OCO)	EXEC CICS SPI commands for FEPI	-	03
DFHEIQTM	CSECT (OCO)	EXEC inquire/discard for autinstmodel	-	03
DFHEIQTR	CSECT (OCO)	EXEC inquire/set for trace	-	03
DFHEIQTS	CSECT (OCO)	EXEC inquire for TS queues	-	03
DFHEIQUE	CSECT (OCO)	EXEC inquire for exit programs	-	03
DFHEIQVT	CSECT	EXEC inquire/set for VTAM and autoinstall	-	03
DFHEIRET	Macro	EXEC interface epilog macro	11	-
DFHEIS	Macro	EXEC interface storage	11	-
DFHEISDS	DSECT	EXEC interface storage definition	11	-
DFHEISEI	DSECT	EXEC interface structure entry I/F	OS	-
DFHEISO	CSECT (OCO)	Sockets Domain API	-	03
DFHEISP	CSECT (OCO)	EXEC interface syncpoint processor	-	03
DFHEISR	CSECT (OCO)	EXEC interface service routines	-	03
DFHEISRA	DSECT	EISR parameter list	OS	-
DFHEISRM	Macro	EISR request	OS	-
DFHEISRT	CSECT (OCO)	EISR trace interpretation data	-	03
DFHEISTG	Macro	EXEC interface storage start macro	11	-
DFHEITAB	CSECT	Translator table (basic commands)	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHEITAL	Other	Cataloged procedure to translate, assemble, and link-edit assembler-language application programs	18	-
DFHEITBS	CSECT	Translator table (special commands)	OS	03
DFHEITCU	CSECT	RDO offline LD table	OS	03
DFHEITDL	Other	Cataloged procedure to translate, compile, and link-edit C/370 application programs	18	-
DFHEITHG	CSECT	EXEC interface hired gun lookup table	OS	03
DFHEITMT	CSECT	Command language table for CEMT	OS	03
DFHEITOT	CSECT	Command language table for CEOT	OS	03
DFHEITPL	Other	Cataloged procedure to translate, compile, and link-edit PL/I application programs	18	-
DFHEITS	CSECT	Temporary storage exec layer	-	03
DFHEITSP	CSECT	Language definition table	OS	03
DFHEITRD	DSECT	Trace point IDs for DFHETC	OS	-
DFHEITST	CSECT	CEST language definition table	OS	03
DFHEITSZ	CSECT (OCO)	EXEC CICS language definition table	-	03
DFHEITTR	CSECT	EXEC interface lookup table	OS	03
DFHEITT2	CSECT	EXEC interface level 2 lookup table	OS	03
DFHEITUT	Source	Definition of EIP trace entries	OS	-
DFHEITVL	Other	Cataloged procedure to translate, compile, and link-edit VS COBOL II application programs	18	-
DFHEIUOW	DSECT	EXEC inquire/set uow, or inquire uoqenq uowlink and uowdsnfail	-	03
DFHEIUS	DSECT	EXEC interface storage - USER part	OS	-
DFHEIVAR	DSECT	COBOL working storage	C2	-
DFHEIWB	CSECT	CWI Language Table and EXEC Layer	-	03
DFHEJBB	CSECT	EJ Bean Browse	-	03
DFHEJBBT	CSECT		-	03
DFHEJBG	CSECT	EJ Bean General	-	03
DFHEJBGT	CSECT		-	03
DFHEJC	CSECT	EXEC interface for journaling	OS	03
DFHEJCB	CSECT	EJ CorbaServer Browse	-	03
DFHEJCBT	CSECT		-	03
DFHEJCG	CSECT	EJ CorbaServer General	-	03
DFHEJCGT	CSECT		-	03
DFHEJCPT	CSECT		-	03
DFHEJDB	CSECT	EJ DJar Browse	-	03
DFHEJDBT	CSECT		-	03
DFHEJDG	CSECT	EJ DJar General	-	03
DFHEJDGT	CSECT		-	03
DFHEJDI	CSECT	DFHEJDI Design EJ Domain EJDI gate functions	-	03
DFHEJDIT	CSECT		-	03
DFHEJDM	CSECT	EJ Domain Functions	-	03
DFHEJDND	Sample	Distinguished name URM	-	19
DFHEJDNH	Sample		-	19
DFHEJDNL	Sample	Distinguished name URM	-	19
DFHEJDNO	Sample	Distinguished name URM	-	19
DFHEJDNX	Sample	Distinguished name URM	-	03
DFHEJDN1	Sample	Distinguished name URM	-	19
DFHEJDN2	Sample	CICS-supplied C-language version of DFHEJDNX	-	19
DFHEJDU	CSECT	EJ domain EJDU gate functions	-	03
DFHEJDUF	CSECT	Dump interpretation for EJ Domain	-	03
DFHEJDUT	CSECT		-	03
DFHEJECT	Macro	Page eject/space option	11	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHEJGE	CSECT	EJ General Operations	-	03
DFHEJGET	CSECT		-	03
DFHEJIO	CSECT	EJ Domain Functions	-	03
DFHEJIOT	CSECT		-	03
DFHEJIT	CSECT	EJ Transaction Functions	-	03
DFHEJJO	CSECT	EJ Domain Functions	-	03
DFHEJJOT	CSECT		-	03
DFHEJMI	CSECT	EJMI CDURUN and Gate Module	-	03
DFHEJMID	CSECT	Message Numbers for the EJ Domain	11	-
DFHEJMIT	CSECT		-	03
DFHEJOB	CSECT	Object Store Browse	-	03
DFHEJOBT	CSECT		-	03
DFHEJOS	CSECT	Object Store Program	-	03
DFHEJOST	CSECT		-	03
DFHEJRDS	CSECT		11	07
DFHEJST	CSECT	EJ Domain - Statistics (STST) gate	-	03
DFHEJTBB	CSECT		-	03
DFHEJTBG	CSECT		-	03
DFHEJTB1	CSECT		-	03
DFHEJTCB	CSECT		-	03
DFHEJTCG	CSECT		-	03
DFHEJTC1	CSECT		-	03
DFHEJTDB	CSECT		-	03
DFHEJTDG	CSECT		-	03
DFHEJTDM	CSECT		-	03
DFHEJTD1	CSECT		-	03
DFHEJTGE	CSECT		-	03
DFHEJTID	Macro	Trace Points for the EJ Domain	11	-
DFHEJTIO	CSECT		-	03
DFHEJTIT	CSECT		-	03
DFHEJTJO	CSECT		-	03
DFHEJTRI	CSECT	EJ Trace Domain interpretation	-	03
DFHEJUPA	Macro	EJ XRSINDI Overlay	11	-
DFHEJXDF	CSECT	Transaction Dump - JRAS dump info	-	03
DFHEKC	CSECT	EXEC interface for task control	OS	03
DFHELII	CSECT	EXEC interface link-edit stub for C/370 application programs	OS	03
DFHEMBA	CSECT	EM Domain - EMBA gate functions	-	03
DFHEMBR	CSECT	EM Domain - EMBR gate functions	-	03
DFHEMBRT	CSECT		-	03
DFHEMDM	CSECT	EM Domain - DMDM gate functions	-	03
DFHEMDUF	CSECT	DFHEMDUF Design	-	03
DFHEMEM	CSECT	EM Domain - EMEM gate functions	-	03
DFHEMEMT	CSECT		-	03
DFHEMEX	CSECT	EXEC interface for ME domain	-	03
DFHEMPID	CSECT	Monitoring emp-ids	11	-
DFHEMS	CSECT	EXEC interface for BMS	OS	03
DFHEMT00	CSECT	Master terminal - CEMT/CEOT/CEST program	OS	03
DFHEMT01	CSECT	Master terminal - control module	OS	03
DFHEMT02	CSECT	Master terminal - initialization	OS	03
DFHEMT11	CSECT	Master terminal - atomization	OS	03
DFHEMT12	CSECT	Master terminal - argument analysis	OS	03
DFHEMT19	CSECT	Master terminal - command analysis	OS	03
DFHEMT20	CSECT	Master terminal - table analysis	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHEMT21	CSECT	Master terminal - keyword analysis	OS	03
DFHEMT22	CSECT	Master terminal - special case code	OS	03
DFHEMT23	CSECT	Master terminal - plist generation	OS	03
DFHEMT27	CSECT	Master terminal - spelling correction	OS	03
DFHEMT50	CSECT	Master terminal - special displays	OS	03
DFHEMT51	CSECT	Master terminal - display extraction	OS	03
DFHEMT52	CSECT	Master terminal - syntax display	OS	03
DFHEMT53	CSECT	Master terminal - utilities	OS	03
DFHEMT54	CSECT	Master terminal - further utilities	OS	03
DFHEMT55	CSECT	Master terminal - fulists	OS	03
DFHEMT56	CSECT	Master terminal - execution interface	OS	03
DFHEMTRI	CSECT	DFHEMTRI Design	-	03
DFHEND	Macro	Generate END statement	11	-
DFHENV	Macro	CICS environment service request	OS	-
DFHEOP	CSECT (OCO)	EXEC interface for write operator	-	03
DFHEPC	CSECT	EXEC interface for program control	-	03
DFHEPILO	Macro	Free automatic storage application epilog	OS	-
DFHEPMAP	CSECT	PL/I translator - advanced code generation functions	OS	03
DFHEPMEE	CSECT	PL/I translator - error editor	OS	03
DFHEPMPP	CSECT	PL/I translator - primary code generation functions	OS	03
DFHEPMSP	CSECT	PL/I translator - input scanner	OS	03
DFHEPM02	CSECT	PL/I translator - initialization	OS	03
DFHEPM07	CSECT	PL/I translator - options card	OS	03
DFHEPM08	CSECT	PL/I translator - check options	OS	03
DFHEPM10	CSECT	PL/I translator - analyze program	OS	03
DFHEPM11	CSECT	PL/I translator - atomization	OS	03
DFHEPM14	CSECT	PL/I translator - read input	OS	03
DFHEPM17	CSECT	PL/I translator - generate output	OS	03
DFHEPS	CSECT	System spooling interface stub	OS	03
DFHERDUF	CSECT (OCO)	SDUMP error message index processor	-	03
DFHERM	CSECT	Resource manager interface (RMI) module	-	03
DFHERMRS	CSECT	ERM resync processor	-	03
DFHERMSP	CSECT	ERM syncpoint processor	-	03
DFHESC	CSECT	EXEC interface for storage control	OS	03
DFHESE	CSECT (OCO)	EXEC interface for query security	-	03
DFHESN	CSECT (OCO)	EXEC interface for signon and sign-off	-	03
DFHESP00	CSECT	RDO - CEDA/CEDB/CEDC program	OS	03
DFHESP01	CSECT	RDO - CEDA control module	OS	03
DFHESP02	CSECT	RDO - CEDA initialization	OS	03
DFHESP11	CSECT	RDO - CEDA atomization	OS	03
DFHESP12	CSECT	RDO - CEDA argument analysis	OS	03
DFHESP19	CSECT	RDO - CEDA command analysis	OS	03
DFHESP20	CSECT	RDO - CEDA table analysis	OS	03
DFHESP21	CSECT	RDO - CEDA keyword analysis	OS	03
DFHESP22	CSECT	RDO - CEDA special case code	OS	03
DFHESP23	CSECT	RDO - CEDA plist generation	OS	03
DFHESP26	CSECT	RDO - CEDA message editor	OS	03
DFHESP27	CSECT	RDO - CEDA spelling correction	OS	03
DFHESP50	CSECT	RDO - CEDA special displays	OS	03
DFHESP51	CSECT	RDO - CEDA display extraction	OS	03
DFHESP52	CSECT	RDO - CEDA syntax display	OS	03
DFHESP53	CSECT	RDO - CEDA utilities	OS	03
DFHESP54	CSECT	RDO - CEDA further utilities	OS	03
DFHESP55	CSECT	RDO - CEDA fulists	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHESZ	CSECT (OCO)	EXEC CICS API commands for FEPI	-	03
DFHETC	CSECT	EXEC interface for terminal control	OS	03
DFHETCB	Macro	EXEC terminal control block macro	OS	-
DFHETD	CSECT	EXEC interface for transient data	OS	03
DFHETL	CSECT	LU6.2 EXEC interface stub	OS	03
DFHETR	CSECT	EXEC interface for trace control	OS	03
DFHETRX	CSECT (OCO)	EXEC interface for enter tracenum, monitor	-	03
DFHEXAI	CSECT	Link-edit stub for assembler-language programs using CSD offline extract function	OS	03
DFHEXCI	CSECT	Link-edit stub for COBOL programs using CSD offline extract function	OS	03
DFHEXDUF	CSECT (OCO)	EXCI dump formatting routine	-	03
DFHEXI	CSECT	Terminal exceptional input program	OS	03
DFHEXLE	CSECT		OS	03
DFHEXLI	CSECT	EXCI stub	11	-
DFHEXMAB	CSECT	Translators - default argument text build	OS	03
DFHEXMAN	CSECT	Translators - statement syntax analysis	OS	03
DFHEXMG1	CSECT	Translators - EXEC DLI code generator	OS	03
DFHEXMG2	CSECT	Translators - EXEC CICS code generator	OS	03
DFHEXMG3	CSECT	Translators - EXEC CICS GDS code generator	OS	03
DFHEXMG4	CSECT	Translators - EXEC EXCI code generator	OS	03
DFHEXMG5	CSECT	Translators - CICSPlex SM EXEC CICS command code generator	-	03
DFHEXMKW	CSECT	Translators - keyword analysis	OS	03
DFHExphE	CSECT	Translators - fatal error handler	OS	03
DFHEXMS1	CSECT	Translators - DL/I WHERE operand code generator	OS	03
DFHEXMS2	CSECT	Translators - EXEC CICS special case code generator	OS	03
DFHEXMS3	CSECT	Translators - EXEC CICS GDS special case code generator	OS	03
DFHEXMS4	CSECT	Translators - EXEC EXCI special case code generator	OS	03
DFHEXMS5	CSECT	Translators - EXEC EXCI special case code generator for CICSPlex SM	-	03
DFHEXMTD	CSECT	Translators - temporaries declaration	OS	03
DFHEXMTG	CSECT	Translators - EXEC trigger detection	OS	03
DFHEXMXK	CSECT	Translators - syntax checker	OS	03
DFHEXMXM	CSECT	Translators - syntax check error messages	OS	03
DFHEXMXS	CSECT	Translators - syntax check control module	OS	03
DFHEXM00	CSECT	Translators - control module	OS	03
DFHEXM01	CSECT	Translators - control module	OS	03
DFHEXM05	CSECT	Translators - PARM analysis	OS	03
DFHEXM06	CSECT	Translators - process single option	OS	03
DFHEXM09	CSECT	Translators - print options	OS	03
DFHEXM12	CSECT	Translators - match brackets	OS	03
DFHEXM13	CSECT	Translators - diagnosis	OS	03
DFHEXM15	CSECT	Translators - I/O module	OS	03
DFHEXM16	CSECT	Translators - conversions	OS	03
DFHEXM18	CSECT	Translators - insert in I/O buffer	OS	03
DFHEXM25	CSECT	Translators - print xref	OS	03
DFHEXM27	CSECT	Translators - spelling correction	OS	03
DFHEXPI	CSECT	Link-edit stub for PL/I programs using CSD offline extract function	OS	03
DFHEXTAL	Other	Cataloged procedure to translate, assemble, and link-edit Assembler- language application programs (EXCI)	18	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHEXTDL	Other	Cataloged procedure to translate, compile, and link-edit C/370 application programs (EXCI)	18	-
DFHEXTM	Macro	Dummy macro for DOS compatibility	OS	-
DFHEXTPL	Other	Cataloged procedure to translate, compile, and link-edit PL/I application programs (EXCI)	18	-
DFHEXTRI	Macro	EXCI trace interpretation routine	-	03
DFHEXTVL	Other	Cataloged procedure to translate, compile, and link-edit VS COBOL II application programs (EXCI)	18	-
DFHFAUED	DSECT		-	11
DFHFBPDS	DSECT	File buffer pool control block	OS	-
DFHFCAT	CSECT	File control catalog manager	OS	03
DFHFCATA	DSECT	FCAT parameter list	OS	-
DFHFCATM	Macro	FCAT request	OS	-
DFHFCATT	CSECT	FCAT translate tables	OS	03
DFHFCBD	CSECT	File control BDAM request processor	OS	03
DFHFCCA	CSECT (OCO)	File control RLS control ACB manager	-	03
DFHFCCAT	CSECT (OCO)	FCCA translate tables	-	03
DFHFCCRT	CSECT		-	03
DFHFCCTT	CSECT		-	03
DFHFCCUT	CSECT		-	03
DFHFCDL	CSECT	File control CFDT Load	-	03
DFHFCDN	CSECT (OCO)	File control DSN block manager	-	03
DFHFCDNA	DSECT	FCDN parameter list	OS	-
DFHFCDNM	Macro	FCDN request	OS	-
DFHFCDNT	CSECT (OCO)	FCDN translate tables	-	03
DFHFCDO	CSECT	File control CFDT Open/Close	-	03
DFHFCDR	CSECT	FC CF data table request handler	-	03
DFHFCDST	CSECT		-	03
DFHFCDTS	CSECT (OCO)	Shared data table request program	-	03
DFHFCDTX	CSECT (OCO)	File control shared data table function ship program	-	03
DFHFCDU	CSECT	File control CFDT Recovery Control	-	03
DFHFCDUF	CSECT (OCO)	File control SDUMP formatter	-	03
DFHFCDUT	CSECT		-	03
DFHFCDW	CSECT	File control CFDT Recovery Control	-	03
DFHFCDY	CSECT	File control CFDT Recovery Resynchronization	-	03
DFHFCDYT	CSECT		-	03
DFHFCEDS	DSECT	File control EXEC argument list	11	-
DFHFCES	CSECT (OCO)	File control ENF servicer	-	03
DFHFCFL	CSECT (OCO)	File control FRAB/FLAB processor	-	03
DFHFCFLA	DSECT	FCFL parameter list	OS	-
DFHFCFLI	Macro	File control test file user	OS	-
DFHFCFLM	Macro	FCFL request	OS	-
DFHFCFLT	CSECT	FCFL translate tables	-	03
DFHFCFR	CSECT	File control file request handler	OS	03
DFHFCFRA	DSECT	FCFR parameter list	OS	-
DFHFCFRM	Macro	FCFR request	OS	-
DFHFCFRT	CSECT	FCFR trace interpretation data	OS	03
DFHFCFS	CSECT	File control file state program	OS	03
DFHFCFSA	DSECT	FCFS parameter list	OS	-
DFHFCFSM	Macro	FCFS request	OS	-
DFHFCFST	CSECT	FCFS translate tables	OS	03
DFHFCINA	DSECT	FCIN parameter list	OS	-
DFHFCINM	Macro	FCIN request	OS	-
DFHFCINT	CSECT	FCIN translate tables	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHFCIN1	CSECT	File control initialization program 1	OS	03
DFHFCIN2	CSECT	File control initialization program 2	OS	03
DFHFCIR	CSECT (OCO)	File control initialize recovery module	-	03
DFHFCL	CSECT	File control VSAM LSR pool processor	OS	03
DFHFCLF	CSECT (OCO)	File control logger failures	-	03
DFHFCLGD	CSECT	File control part of log record	11	-
DFHFCLJ	CSECT (OCO)	File control logging and journaling	-	03
DFHFCLJA	DSECT	FCLJ parameter list	OS	-
DFHFCLJM	Macro	FCLJ request	OS	-
DFHFCLJT	CSECT	FCLJ translate tables	-	03
DFHFCLTD	DSECT	File control logger user token	11	-
DFHFCM	CSECT	File control VSAM KSDS base open/close	OS	03
DFHFCMT	CSECT (OCO)	File control table manager	-	03
DFHFCMTA	DSECT	FCMT parameter list	OS	-
DFHFCMTM	Macro	FCMT request	OS	-
DFHFCMTT	CSECT (OCO)	FCMT translate tables	-	03
DFHFCN	CSECT	File control open/close program	OS	03
DFHFCNC	Source	File control - close request	OS	-
DFHFCNO	Source	File control - open request	OS	-
DFHFCNQ	CSECT (OCO)	File control non-RLS lock handler	-	03
DFHFCOR	CSECT (OCO)	File control RLS offsite recovery completion	-	03
DFHFCQI	CSECT	File control - VSAM RLS quiesce initiation module	-	03
DFHFCQIT	DSECT	FCQI translate tables	-	03
DFHFCQR	CSECT (OCO)	File control - VSAM RLS quiesce receive module	-	03
DFHFCQRT	DSECT	FCQR translate tables	-	03
DFHFCQS	CSECT (OCO)	File control - VSAM RLS quiesce send module	-	03
DFHFCQST	DSECT	FCQS translate tables	-	03
DFHFCQT	CSECT (OCO)	File control - VSAM RLS quiesce - common system transaction	-	03
DFHFCQU	CSECT (OCO)	File control - VSAM RLS quiesce process module	-	03
DFHFCQUT	DSECT	FCQU translate tables	-	03
DFHFCQX	CSECT (OCO)	File control - VSAM RLS quiesce exit module	-	03
DFHFCRC	CSECT (OCO)	File control recovery control	-	03
DFHFCRD	CSECT (OCO)	File control VSAM RLS post server-failure recovery	-	03
DFHFCRF	CSECT	File control Remote Interface	-	03
DFHFCRFA	CSECT	FCRF interface parameter area	11	-
DFHFCRFM	Macro	DFHFCRF interface macro	OS	-
DFHFCRFT	CSECT		-	03
DFHFCRL	CSECT (OCO)	File control VSAM SHRCTL block manager	-	03
DFHFCRLA	DSECT	FCRL parameter list	OS	-
DFHFCRLM	Macro	FCRL request	OS	-
DFHFCRLT	CSECT (OCO)	FCRL translate tables	-	03
DFHFCRO	CSECT (OCO)	File control VSAM RLS open/close processor	-	03
DFHFCRP	CSECT	File control restart program	OS	03
DFHFCRPA	DSECT	FCRP parameter list	OS	-
DFHFCRPM	Macro	FCRP request	OS	-
DFHFCRPT	CSECT	FCRP translate tables	OS	03
DFHFCRR	CSECT (OCO)	File control RLS restart program	-	03
DFHFCRRT	CSECT	FCRR translate tables	-	03
DFHFCRS	CSECT (OCO)	File control RLS record management program	-	03
DFHFCRV	CSECT (OCO)	File control RLS VSAM interface program	-	03
DFHFCSD	CSECT	File control shutdown program	OS	03
DFHFCSDA	DSECT	FCSD parameter list	OS	-
DFHFCSDM	Macro	FCSD request	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHFCSDS	DSECT	File control static storage	11	-
DFHFCSDT	CSECT	FCSD translate tables	OS	03
DFHFCST	CSECT	File control statistics program	OS	03
DFHFCSTA	DSECT	FCST parameter list	OS	-
DFHFCSTM	Macro	FCST request	OS	-
DFHFCSTT	CSECT	FCST translate tables	OS	03
DFHFCT	Macro	File control table	11	-
DFHFCTDS	DSECT	File control table entry	11	-
DFHFCTRN	Symbolic	File control trace, message, and catalog	OS	-
DFHFCTSP	Macro	FCT shared resources control block generator	11	-
DFHFCTSR	DSECT	FCT shared resources control block	11	-
DFHFCU	CSECT	File open utility program	OS	03
DFHFCVR	CSECT	File control VSAM interface program	OS	03
DFHFCVS	CSECT	File access VSAM request processor	OS	03
DFHFCWS	Macro	File control work areas	OS	-
DFHFCXDF	CSECT	DU domain - transaction dump formatter for file-related areas	OS	03
DFHFEP	CSECT	Field engineering program	OS	03
DFHFIOA	DSECT	File input/output area	OS	-
DFHFLABD	DSECT	File lasting access block	OS	-
DFHFMH	Macro	Function management header	OS	-
DFHFMHDS	DSECT	Function management header	11	-
DFHFMIDS	Symbolic	Function and module identifiers	11	-
DFHFORMS	CSECT		-	03
DFHFRABD	DSECT	File request anchor block	OS	-
DFHFRDUF	CSECT (OCO)	File control recoverable work elements SDUMP formatter	-	03
DFHFRTED	DSECT	File request thread element	OS	-
DFHFTDUF	CSECT (OCO)	Print feature 'FT' keyword processor	-	03
DFHFTTRI	CSECT (OCO)	Offline TR entries trace interpretation	OS	03
DFHGCAA	CSECT	Language Environment - get common anchor area	OS	03
DFHGDEFS	Symbolic	CICS global symbol definitions	11	-
DFHGMM	CSECT	VTAM LU startup message	OS	03
DFHHASH	Macro	Locate TCTTE entries	OS	-
DFHHLPDS	DSECT	DL/I interface block	D3	-
DFHHLPDS	Macro	CICS-IMS HLPI control blocks	OS	08
DFHHMDCD	DSECT	Handle manager table block	OS	-
DFHHPSVC	CSECT	HPO type 6 SVC	OS	03
DFHIC	Macro	Time service request	11	-
DFHICDUF	CSECT (OCO)	Interval control SDUMP formatter	-	03
DFHICEDS	DSECT	Interval control element	11	-
DFHICP	CSECT	Interval control program	OS	03
DFHICRC	CSECT	Interval control recovery module	-	03
DFHICUED	CSECT	EXEC argument list for Interval Control	11	-
DFHICXM	CSECT	AP domain - bind, inquire, and release facility IC functions	OS	03
DFHICXMA	DSECT	ICXM parameter list	OS	-
DFHICXMM	Macro	ICXM request	OS	-
DFHICXMT	CSECT	ICXM translate tables	OS	03
DFHIEDM	CSECT	IE Domain Initialization/Termination	-	03
DFHIEDUF	CSECT	IE Domain System Dump Formatting	-	03
DFHIEIE	CSECT	IP ECI Listener	-	03
DFHIEIEA	CSECT	IEIE interface parameter area	OS	-
DFHIEIEM	Macro	DFHIEIE interface macro	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHIEIET	CSECT		-	03
DFHIEP	CSECT		-	03
DFHIETRI	CSECT	IP ECI Domain Trace Interpretation	-	03
DFHIEXM	CSECT		-	03
DFHIHFSA	CSECT		-	02
DFHIHFS0	CSECT		-	02
DFHIHFS1	CSECT		-	02
DFHIICP	CSECT	IIOP Command Processor	-	03
DFHIIDM	CSECT	II Domain Initialization/Termination	-	03
DFHIIDUF	CSECT	II Domain System Dump Formatting	-	03
DFHIILST	CSECT		-	03
DFHIIMM	CSECT	DFHIIMM Design II domain - IIMM gate functs.	-	03
DFHIIMT	CSECT		-	03
DFHIIP	CSECT	BMS non-3270 input mapping	OS	-
DFHIIPA$	CSECT	BMS non-3270 input mapping (standard)	OS	03
DFHIIP1$	CSECT	BMS non-3270 input mapping (full)	OS	03
DFHIIRDS	DSECT		11	07
DFHIIRH	DSECT	IIOP Request Handler	-	03
DFHIIRHT	DSECT		-	03
DFHIIRP	CSECT	IIOP Request Processor	-	03
DFHIIRPT	DSECT		-	03
DFHIIRQ	CSECT	DFHIIRQ Design	-	03
DFHIIRQT	CSECT		-	03
DFHIIRR	CSECT	IIOP Request Receiver	-	03
DFHIIRRS	CSECT		-	03
DFHIIRRT	CSECT		-	03
DFHIIST	CSECT	II Domain - Statistics (STST) gate	-	03
DFHIITRI	CSECT	IIOP Domain Trace Interpretation	-	03
DFHIIURH	CSECT		-	08
DFHIIXM	CSECT	IIOP Attach Client	-	03
DFHIJVME	Other	Customize a member of the SDFHENV library	02	-
DFHIJVMJ	Other		02	-
DFHILG1	Other	Define logstream CF structures to MVS logger	02	-
DFHILG2	Other	Define logstream models for system log streams	02	-
DFHILG3	Other	Define logstream models for individual CICS region	02	-
DFHILG4	Other	Define specific logstream for log of logs	02	-
DFHILG5	Other		02	-
DFHILG6	Other		02	-
DFHILG7	Other		02	-
DFHIMSDS	DSECT	ISC message inserts	11	-
DFHINDAP	CSECT	Indoubt tool	-	03
DFHINDSP	CSECT	Indoubt tool syncpoint processor	-	03
DFHINDT	CSECT	Indoubt tool	-	03
DFHINST	Other	TSO CLIST to generate installation jobs	02	-
DFHINSTA	Other	JCL to create an additional target zone, CSI, and set of target libraries	02	-
DFHINSTJ	Other	JCL to RECEIVE, APPLY, and ACCEPT the Japanese language feature	02	-
DFHINST1	Other	JCL to allocate and catalog CICS target and distribution libraries	02	-
DFHINST2	Other	JCL to allocate and catalog CICS RELFILE data sets	02	-
DFHINST3	Other	JCL to allocate and catalog CICS SMP/E data sets	02	-
DFHINST4	Other	JCL to initialize CICS SMP/E data sets	02	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHINST5	Other	JCL to RECEIVE the CICS base-level function SYSMOD	02	-
DFHINST6	Other	JCL to APPLY and ACCEPT the CICS base- level function SYSMOD	02	-
DFHINTRU	CSECT	Indoubt tool task related user exit	-	03
DFHIONCD	Other	Replace DDDEFS for Language Environment or TCP/IP libraries in SMP/E target zone	02	-
DFHIONCL	Other	Relink-edit DFHRPRP load module outside SMP/E	02	-
DFHIPCSP	Other	IPCS parmlib imbed member for DFHPDxxxx	-	15
DFHIPDUF	CSECT (OCO)	SDUMP formatter for kernel stack internal procedures	-	03
DFHIPUBS	Other		02	-
DFHIR	Macro	Interregion request	-	11
DFHIRP	CSECT	Interregion communication program	OS	03
DFHIRPAD	Source	IRC dynamic add of connections routines	OS	-
DFHIRPC	Source	IRC connect and disconnect routines	OS	-
DFHIRPCL	Source	IRC clear and logoff routines	OS	-
DFHIRPD	Macro	IRC program internal control blocks	11	-
DFHIRPL	Source	IRC logon routines	OS	-
DFHIRPM	Source	IRC subroutines	OS	-
DFHIRPQ	Source	IRC in-service and quiesce routines	OS	-
DFHIRPR	Source	IRC recovery routines	OS	-
DFHIRPS	Source	IRC subroutines	OS	-
DFHIRPSP	Source	IRC SRB processor	OS	-
DFHIRPSW	Source	IRC switch and pull routines	OS	-
DFHIRRDS	Macro	Interregion session recovery data stream	11	-
DFHIRRXD	Sample	IRC XCF retry DIE subroutine	OS	-
DFHIRRXP	Sample	IRC XCF termination subroutine	OS	-
DFHIRRXS	Sample	IRC XCF SRB processor	OS	-
DFHIRSDS	DSECT	Interregion subsystem control blocks	11	-
DFHIRW10	CSECT	IRC work delivery exit program	OS	03
DFHIS	Macro	ISC request	OS	-
DFHISCRQ	Macro	ISC request parameter list	11	-
DFHISMKD	Other		02	-
DFHISP	CSECT	Intersystem communication program	OS	03
DFHISTAR	Other	JCL to invoke DFHINST	02	-
DFHIVPBT	Other	IVP (batch) to verify CICS startup	02	-
DFHIVPDB	Other	IVP to verify CICS running with DBCTL	02	-
DFHIVPOL	Other	IVP (online) to verify CICS, without DL/I	02	-
DFHJC	Macro	Journal service request	OS	-
DFHJCA	Macro	Journal control area definition	11	-
DFHJCADS	DSECT	Journal control area	11	-
DFHJCDLG	CSECT	Autocall SCEEOBJ	-	03
DFHJCDLL	CSECT	Autocall SCEEOBJ	-	03
DFHJCIMP	CSECT		-	20
DFHJCJCA	DSECT	JCJC parameter list	OS	-
DFHJCJCM	Macro	JCJC request	OS	-
DFHJCJCT	CSECT	JCJC trace interpretation data	OS	03
DFHJCJCX	Macro	JCJC request (XPI)	11	-
DFHJCJCY	DSECT	JCJC parameter list (XPI)	11	-
DFHJCP	CSECT	Journal control program	-	03
DFHJCR	Macro	Journal control record	11	-
DFHJCSTC	CSECT		-	03
DFHJHPA@	CSECT		-	03
DFHJHPAT	Sample	Java Hotpooling Pre-Call URM	-	19

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHJUP	CSECT	Journal control print utility	OS	03
DFHJVCV@	CSECT		-	03
DFHJVMA@	CSECT	Autocall SCEEOBJ	-	03
DFHJVMAT	Sample	CICS JVM Interface user replaceable module	-	19
DFHJVMPR	Other		-	09
DFHJVMPS	Other		-	09
DFHJVTRI	CSECT		-	03
DFHKC	Macro	Task service request	11	-
DFHKCQ	CSECT	Transaction manager - secondary requests	OS	03
DFHKCRP	CSECT	Task control restart program	OS	03
DFHKCSC	CSECT	DFHKCQ chain scanning for discard	OS	03
DFHKCSCA	DSECT	KCSC parameter list	OS	-
DFHKCSCM	Macro	KCSC request	OS	-
DFHKCSCT	CSECT	KCSC trace interpretation data	OS	03
DFHKCSP	CSECT	Task SRB control program	OS	03
DFHKEALI	Macro	KE domain - label alignment	OS	-
DFHKEAR	CSECT (OCO)	KE domain - MVS ARM support services	-	03
DFHKEARA	DSECT	KEAR parameter list	OS	-
DFHKEARM	Macro	KEAR request	OS	-
DFHKEART	CSECT (OCO)	KEAR trace interpretation data	-	03
DFHKEDCL	CSECT (OCO)	KE domain - domain call request handler	-	03
DFHKEDD	CSECT (OCO)	KE domain - domain definition services	-	03
DFHKEDDA	DSECT	KEDD parameter list	OS	-
DFHKEDDM	Macro	KEDD request	OS	-
DFHKEDDT	CSECT (OCO)	KEDD trace interpretation data	-	03
DFHKEDRT	CSECT (OCO)	KE domain - domain return request handler	-	03
DFHKEDS	CSECT (OCO)	KE domain - dispatcher interfaces	-	03
DFHKEDSA	DSECT	KEDS parameter list	OS	-
DFHKEDSI	Macro	KE domain - optimize kernel path lengths	OS	-
DFHKEDSM	Macro	KEDS request	OS	-
DFHKEDST	CSECT (OCO)	KEDS trace interpretation data	-	03
DFHKEDSX	Macro	KEDS request	11	-
DFHKEDSY	CSECT	KEDS parameter list	11	-
DFHKEDUF	CSECT (OCO)	SDUMP formatter for KE domain	-	03
DFHKEEDA	CSECT (OCO)	KE domain - execute deferred abend	-	03
DFHKEENV	Macro	KE domain - declare/switch environment	11	-
DFHKEGD	CSECT (OCO)	KE domain - global data services	-	03
DFHKEGDA	DSECT	KEGD parameter list	OS	-
DFHKEGDM	Macro	KEGD request	OS	-
DFHKEGDT	CSECT (OCO)	KEGD trace interpretation data	-	03
DFHKEIN	CSECT (OCO)	KE domain - initialization	-	03
DFHKEINA	DSECT	KEIN parameter list	OS	-
DFHKEINM	Macro	KEIN request	OS	-
DFHKEINT	CSECT (OCO)	KEIN trace interpretation data	-	03
DFHKELCL	CSECT (OCO)	KE domain - LIFO push simulation	-	03
DFHKELOC	CSECT (OCO)	SDUMP routine for locating domain anchors	-	03
DFHKELRT	CSECT (OCO)	KE domain - LIFO return/pop simulation	-	03
DFHKEMD	Macro	KE domain - domain/subroutine prolog code	OS	-
DFHKEPUB	DSECT	KE domain - some control blocks	OS	-
DFHKERCD	CSECT (OCO)	KE domain - kernel error data construction	-	03
DFHKERER	CSECT (OCO)	KE domain - record error routine	-	03
DFHKERET	CSECT (OCO)	KE domain - reset address service	-	03
DFHKERKE	CSECT (OCO)	KE domain - KERNERROR response handler	-	03
DFHKERN	Macro	KE domain - generate call to kernel	11	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHKERPC	CSECT (OCO)	KE domain - recovery percolation	-	03
DFHKERRI	CSECT (OCO)	KE domain - recovery invocation	-	03
DFHKERRQ	CSECT (OCO)	KE domain - recovery request service	-	03
DFHKERRU	CSECT (OCO)	KE domain - runaway task error handler	-	03
DFHKERRX	CSECT (OCO)	KE domain - recovery exit service	-	03
DFHKESCL	CSECT (OCO)	KE domain - subroutine call handler	-	03
DFHKESFM	CSECT (OCO)	KE domain - disposable segments freemain	-	03
DFHKESGM	CSECT (OCO)	KE domain - new stack segments getmain	-	03
DFHKESIP	CSECT (OCO)	KE domain - system initialization program	-	03
DFHKESRT	CSECT (OCO)	KE domain - subroutine return handler	-	03
DFHKESTP	DSECT	KE domain - kernel stack structure	OS	-
DFHKESTX	CSECT (OCO)	KE domain - kernel ESTAE exit	-	03
DFHKESVC	CSECT (OCO)	KE domain - authorized service routine	-	03
DFHKETA	CSECT (OCO)	KE domain - task reply services	-	03
DFHKETAB	CSECT (OCO)	KE domain - list of domains requiring preinitialization on CICS run	-	03
DFHKETB2	CSECT (OCO)	KE domain - list of domains requiring preinitialization on DFHSTUP run	-	03
DFHKETCB	CSECT (OCO)	KE domain - kernel TCB startup routine	-	03
DFHKETI	CSECT (OCO)	KE domain - timer services	-	03
DFHKETIA	DSECT	KETI parameter list	OS	-
DFHKETIM	Macro	KETI request	OS	-
DFHKETIT	CSECT (OCO)	KETI trace interpretation data	-	03
DFHKETIX	CSECT (OCO)	KE domain - STIMER exit	-	03
DFHKETXR	CSECT	KE ETXR	-	03
DFHKEXM	CSECT (OCO)	KE domain - XM domain services	-	03
DFHKEXMA	DSECT	KEXM parameter list	OS	-
DFHKEXMM	Macro	KEXM request	OS	-
DFHKEXMT	CSECT (OCO)	KEXM trace interpretation data	OS	03
DFHKETRI	CSECT (OCO)	Trace interpreter for KE domain	-	03
DFHLANG	Other	List of National Languages for CICS - alias for MEULANG	10	-
DFHLDDM	CSECT (OCO)	LD domain - initialization/termination	-	03
DFHLDDMI	CSECT (OCO)	LD domain - secondary initialization	-	03
DFHLDDUF	CSECT (OCO)	SDUMP formatter for LD domain	-	03
DFHLDGDS	DSECT	LD domain - global statistics	11	-
DFHLDGDS	DSECT	LD domain - global statistics	C2	07
DFHLDGDS	DSECT	LD domain - global statistics	P2	-
DFHLDLDA	DSECT	LDLD parameter list	OS	-
DFHLDLDM	Macro	LDLD request	OS	-
DFHLDLDT	CSECT (OCO)	LDLD trace interpretation data	-	03
DFHLDLDX	Macro	LDLD request (XPI)	11	-
DFHLDLDY	DSECT	LDLD parameter list (XPI)	11	-
DFHLDLD1	CSECT (OCO)	LD domain - acquire/release/refresh	-	03
DFHLDLD2	CSECT (OCO)	LD domain - define/delete	-	03
DFHLDLD3	CSECT (OCO)	LD domain - general functions	-	03
DFHLDNT	CSECT (OCO)	LD domain - storage notify handler	-	03
DFHLDRDS	DSECT	LD domain - program statistics	11	-
DFHLDRDS	DSECT	LD domain - program statistics	C2	07
DFHLDST	CSECT (OCO)	LD domain - statistics collection	-	03
DFHLDSUA	DSECT	LDSU parameter list	OS	-
DFHLDSUM	Macro	LDSU request	OS	-
DFHLDSUT	CSECT (OCO)	LDSU trace interpretation data	-	03
DFHLDSVC	CSECT (OCO)	LD domain - authorized service routine	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHLDTRI	CSECT (OCO)	Trace interpreter for LD domain	-	03
DFHLEAS	CSECT	ADD SUBPOOL service	-	03
DFHLEDS	CSECT	DELETE SUBPOOL service	-	03
DFHLEDT	CSECT	Transaction Dump service	-	03
DFHLEFM	CSECT	GETMAIN service	-	03
DFHLEFQ	CSECT	Quickcell freemain service	-	03
DFHLEGM	CSECT	GETMAIN service	-	03
DFHLEGQ	CSECT	Quickcell getmain service	-	03
DFHLERO	CSECT	Runtime options service	-	03
DFHLESRV	Macro	CICS Service routine vector	11	-
DFHLETR	CSECT	Trace servicve routine	-	03
DFHLETRM	Macro	LE Trace Service invocation macro	11	-
DFHLFM	Macro	LIFO macro	11	-
DFHLFT	Macro	LIFO trace macro	11	-
DFHLFX	Macro	LIFO stack entry	11	-
DFHLGBAA	DSECT	LGBA parameter list	11	-
DFHLGBAM	Macro	LGBA request	OS	-
DFHLGBAT	DSECT (OCO)	LGBA translate tables	-	03
DFHLGCBT	DSECT	LGCB translate tables	-	03
DFHLGCCA	CSECT (OCO)	LGCC parameter list	OS	-
DFHLGCCM	Macro	LGCC request	OS	-
DFHLGCCT	DSECT (OCO)	LGCC translate tables	-	03
DFHLGDM	CSECT (OCO)	Logger domain - domain initialization	-	03
DFHLGDUF	CSECT (OCO)	Log Manager domain dump formatting	-	03
DFHLGFLD	DSECT	Log Manager log of log format	11	-
DFHLGGFD	DSECT	Log Manager general log format	11	-
DFHLGGL	CSECT (OCO)	Log Manager general log gate module	-	03
DFHLGGLA	CSECT (OCO)	LGGL parameter list	OS	-
DFHLGGLI	CSECT (OCO)	Journal number to name conversion	OS	-
DFHLGGLM	Macro	LGGL request	OS	-
DFHLGGLT	DSECT (OCO)	LGGL translate tables	-	03
DFHLGICV	CSECT (OCO)	LG SSI log record conversion to old format	-	03
DFHLGIGT	DSECT	LG LOGR SSI dataset GET exit	-	03
DFHLGILA	CSECT (OCO)	LG Subsytem exit - lexical analyzer	-	03
DFHLGIMS	CSECT (OCO)	LG Subsytem exit - syntax message composer	-	03
DFHLGIPA	CSECT (OCO)	LG Subsytem exit - parser	-	03
DFHLGIPI	CSECT (OCO)	LG Subsytem exit - parse interface routine	-	03
DFHLGISM	CSECT (OCO)	LG Subsytem exit - parse message exits	-	03
DFHLGJN	CSECT (OCO)	Log Manager journal inventory gate module	-	03
DFHLGJNT	DSECT (OCO)	LGJN translate tables	-	03
DFHLGLBA	CSECT (OCO)	LGLB parameter list	OS	-
DFHLGLBM	Macro	LGLB request	OS	-
DFHLGLBT	DSECT (OCO)	LGLB translate tables	-	03
DFHLGLD	CSECT (OCO)	Log Manager JournalModel gate	-	03
DFHLGLDT	DSECT (OCO)	LGLD translate tables	-	03
DFHLGMSD	CSECT (OCO)	Log Manager MVS SMF log format	11	-
DFHLGMVA	CSECT (OCO)	LGMV parameter list	OS	-
DFHLGMVM	Macro	LGMV request	OS	-
DFHLGMVT	DSECT	LGMV translate tables	-	03
DFHLGPA	CSECT (OCO)	Logger Domain - inquire/set parameters	-	03
DFHLGPAA	CSECT (OCO)	LGPA parameter list	OS	-
DFHLGPAM	Macro	LGPA request	OS	-
DFHLGPAT	DSECT (OCO)	LGPA translate tables	-	03
DFHLGPAX	Macro	Log Manager parameter manager PLIST	11	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHLGPAY	DSECT	Log Manager parameter manager PLIST	11	-
DFHLGQC	CSECT (OCO)	Log Manager RLS cleanup	-	03
DFHLGRDS	CSECT (OCO)	Log Manager journal statistics	11	-
DFHLGRDS	CSECT (OCO)	Log Manager journal statistics	C2	07
DFHLGSC	CSECT (OCO)	Log Manager statistics collection	-	03
DFHLGSDS	CSECT (OCO)	Log Manager logstream statistics	11	-
DFHLGSDS	CSECT (OCO)	Log Manager logstream statistics	C2	07
DFHLGSRA	CSECT (OCO)	LGSR parameter list	OS	-
DFHLGSRT	DSECT (OCO)	LGSR translate tables	OS	03
DFHLGSSI	CSECT (OCO)	Log Manager LOGR SSI dataset exit	-	03
DFHLGST	CSECT (OCO)	Log Manager stream connection gate	-	03
DFHLGSTT	DSECT (OCO)	LGST translate tables	-	03
DFHLGTRI	CSECT (OCO)	Logger - trace interpretation	-	03
DFHLGWFT	DSECT	LGWF translate tables	-	03
DFHLIFO	DSECT	KE domain - LIFO control blocks	OS	-
DFHLILBD	Source	Language interface program language block	OS	-
DFHLILIA	Source	Language interface parameter list	OS	-
DFHLILII	Source	AP domain - Perform goto call to language interface	OS	-
DFHLILIM	Source	Language interface services	OS	-
DFHLILIT	CSECT (OCO)	Language interface trace interpretation data	-	03
DFHLIRET	CSECT (OCO)	Language interface return program	-	03
DFHLITRI	CSECT (OCO)	Language interface trace interpreter	-	03
DFHLIWAD	Source	Language interface work area	OS	-
DFHLI000	Macro		11	-
DFHLLDC	DSECT	Local logical device code table	11	-
DFHLLDLI	DSECT	DLI call level api macro (alias of CALLDLI)	11	-
DFHLMDM	CSECT (OCO)	LM domain - initialization/termination	-	03
DFHLMDS	CSECT (OCO)	LM domain - dispatcher notify handler	-	03
DFHLMDUF	CSECT (OCO)	SDUMP formatter for LM domain	-	03
DFHLMIQ	CSECT (OCO)	LM domain - browse and inquiry	-	03
DFHLMIQA	DSECT	LMIQ parameter list	OS	-
DFHLMIQM	Macro	LMIQ request	OS	-
DFHLMIQT	CSECT (OCO)	LMIQ trace interpretation data	-	03
DFHLMLM	CSECT (OCO)	LM domain - services	-	03
DFHLMLMA	DSECT	LMLM parameter list	OS	-
DFHLMLMI	CSECT		OS	-
DFHLMLMM	Macro	LMLM request	OS	-
DFHLMLMT	CSECT (OCO)	LMLM trace interpretation data	-	03
DFHLMTRI	CSECT (OCO)	Trace interpreter for LM domain	-	03
DFHLNKVS	Other	Cataloged procedure to link-edit CICS programs and application programs	18	-
DFHLOCK	Macro	KE domain - lock/unlock TCB entry	OS	-
DFHLONGN	Other	LD dllload long name conversion	-	03
DFHLPUMD	Other	JCL to RECEIVE and APPLY the DFH$UMOD SMP/E USERMOD	02	-
DFHLSCU	CSECT		-	03
DFHLSTNT	CSECT		-	03
DFHLTRC	CSECT	Local terminal recovery module	-	03
DFHLUC	Macro	LU6.2 service request	OS	-
DFHLUCM	Macro	LU6.2 migration request	OS	-
DFHLUS	Macro	LU6.2 services manager driver macro	OS	-
DFHL2BA	CSECT (OCO)	Log Manager LGBA gate	-	03
DFHL2BL1	CSECT (OCO)	Logger block initialize class procedure	-	03
DFHL2BL2	CSECT (OCO)	Logger block restore current position	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library
DFHL2BS1	CSECT (OCO)	Obtain and initialize BrowseableStream class data	- 03
DFHL2BS2	CSECT (OCO)	Construct a BrowseableStream object and return to caller	- 03
DFHL2BS3	CSECT (OCO)	Destroy a BrowseableStream object	- 03
DFHL2BS4	CSECT (OCO)	Terminate all browseable stream instances known to BrowseableStream class	- 03
DFHL2CB	CSECT (OCO)	Log Manager LGCB gate	- 03
DFHL2CC	CSECT (OCO)	Log Manager LGCC gate	- 03
DFHL2CHA	CSECT (OCO)	Logger chain start browse all procedure	- 03
DFHL2CHE	CSECT (OCO)	Logger chain delete history procedure	- 03
DFHL2CHG	CSECT (OCO)	Logger chain get next chain procedure	- 03
DFHL2CHH	CSECT (OCO)	Logger chain start browse chains procedure	- 03
DFHL2CHI	CSECT (OCO)	Logger chain end browse chains procedure	- 03
DFHL2CHL	CSECT (OCO)	Logger chain end browse all procedure	- 03
DFHL2CHM	CSECT (OCO)	Logger chain move procedure	- 03
DFHL2CHN	CSECT (OCO)	Logger chain browse all get next procedure	- 03
DFHL2CHO	CSECT (OCO)		- 03
DFHL2CHP	CSECT (OCO)		- 03
DFHL2CHR	CSECT (OCO)	Logger chain restore procedure	- 03
DFHL2CHS	CSECT (OCO)	Logger chain set history procedure	- 03
DFHL2CH1	CSECT (OCO)	Logger chain initialize class procedure	- 03
DFHL2CH2	CSECT (OCO)	Logger chain create fresh procedure	- 03
DFHL2CH3	CSECT (OCO)	Logger chain start chain browse procedure	- 03
DFHL2CH4	CSECT (OCO)	Logger chain browse get next procedure	- 03
DFHL2CH5	CSECT (OCO)	Logger chain end chain browse procedure	- 03
DFHL2DM	CSECT (OCO)	Log Manager L2 domain management	- 03
DFHL2DU0	CSECT (OCO)	Log Manager L2_Dump_Formatting_Module	- 03
DFHL2HB	CSECT (OCO)		- 03
DFHL2HSF	CSECT (OCO)	Logger HardStream write MVS retry intro.	- 03
DFHL2HSG	CSECT (OCO)	Logger HardStream read browse cursor	- 03
DFHL2HSJ	CSECT (OCO)	Logger HardStream end browse cursor	- 03
DFHL2HS2	CSECT (OCO)	Logger HardStream connect procedure	- 03
DFHL2HS3	CSECT (OCO)	Logger HardStream disconnect procedure	- 03
DFHL2HS4	CSECT (OCO)	Logger HardStream delete all procedure	- 03
DFHL2HS5	CSECT (OCO)	Logger HardStream delete history procedure	- 03
DFHL2HS6	CSECT (OCO)	Logger HardStream start browse cursor	- 03
DFHL2HS7	CSECT (OCO)	Logger HardStream start read procedure	- 03
DFHL2HS8	CSECT (OCO)	Logger HardStream read block procedure	- 03
DFHL2HS9	CSECT (OCO)	Logger HardStream end read procedure	- 03
DFHL2LB	CSECT (OCO)	Log Manager LGLB gate	- 03
DFHL2MV	CSECT (OCO)	Log Manager LGMV gate	- 03
DFHL2OFI	CSECT (OCO)	Logger object factory initialize procedure	- 03
DFHL2SLE	CSECT (OCO)	Logger system log notify failure method	- 03
DFHL2SLN	CSECT (OCO)	Logger system log open stream method	- 03
DFHL2SL1	CSECT (OCO)	Logger system log initialize class procedure	- 03
DFHL2SR	CSECT (OCO)	Log Manager stream class class declaration	- 03
DFHL2SR1	CSECT (OCO)	Logger stream class initialize class	- 03
DFHL2SR2	CSECT (OCO)	Logger stream class construct procedure	- 03
DFHL2SR3	CSECT (OCO)	Logger stream class destruct procedure	- 03
DFHL2SR4	CSECT (OCO)	Logstream statistics module	- 03
DFHL2SR5	CSECT (OCO)	Logger stream class terminate all procedure	- 03
DFHL2TI2	CSECT (OCO)		- 03
DFHL2TRI	CSECT (OCO)	Log Manager trace interpretation	- 03
DFHL2VP1	CSECT (OCO)	Logger storage manager initialize class	- 03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHL2WF	CSECT (OCO)	Log Manager LGWF gate	-	03
DFHMAPDS	DSECT	BMS map description	OS	-
DFHMAPS	Other	Cataloged procedure to prepare physical and symbolic maps	18	-
DFHMAPT	Other		18	-
DFHMBCDS	DSECT	Transient data buffer control	OS	-
DFHMBMBA	DSECT	File control DFHMBMBI parameter list	OS	-
DFHMBMBI	Macro	File control buffer management inline	OS	-
DFHMCAD	Macro	Map control area	11	-
DFHMCBDS	DSECT	BMS message control block	11	-
DFHMCP	CSECT	BMS mapping control program	OS	-
DFHMCPA$	CSECT	BMS mapping control program (standard)	OS	03
DFHMCPE	CSECT	BMS minimum function mapping control	OS	-
DFHMCPE$	CSECT	BMS mapping control program (minimum)	OS	03
DFHMCPIN	CSECT	BMS input mapping request handler	OS	-
DFHMCPLK	Macro	Linkage to BMS modules	OS	-
DFHMCP1$	CSECT	BMS mapping control program (full)	OS	03
DFHMCRDS	DSECT	BMS message control record	11	-
DFHMCT	Macro	Monitoring control table	11	-
DFHMCTA$	Sample	Monitoring control table for an AOR	19	-
DFHMCTDR	Macro	Monitoring dictionary definition	11	-
DFHMCTDS	Macro	MCT root section definition	11	-
DFHMCTDT	Macro	Transaction monitoring field and dictionary entry definition	11	-
DFHMCTD$	Sample	Monitoring control table for an AOR with DBCTL	19	-
DFHMCTEN	Macro	MCT option macro	11	-
DFHMCTF$	Sample	Monitoring control table for an FOR	19	-
DFHMCTMP	Macro	MCT class macro	11	-
DFHMCTNM	Macro	Monitoring numeric string check	11	-
DFHMCTSE	Macro	MCT option entry generator	11	-
DFHMCTT$	Sample	Monitoring control table for a TOR	19	-
DFHMCT2$	Sample	Monitoring control table	19	03
DFHMCX	CSECT	BMS fast path module	OS	03
DFHMCY	CSECT	Process MAPPINGDEV Requests	OS	03
DFHMDC	Macro	Build C language symbolic description map	11	-
DFHMDCL	Macro	Convert C field names to lowercase	11	-
DFHMDF	Macro	Generate BMS field definition	11	-
DFHMDI	Macro	Generate BMS map definition	11	-
DFHMDX	Macro		11	-
DFHMEACC	CSECT	ME domain - DFHACxxxx message set simplified Chinese version	14	03
DFHMEACE	CSECT	ME domain - DFHACxxxx message set	14	03
DFHMEACK	CSECT (OCO)	ME domain - DFHACxxxx message set	14	03
DFHMEADC	CSECT	ME domain - DFHADxxxx message set simplified Chinese version	14	03
DFHMEADE	CSECT	ME domain - DFHADxxxx message set	14	03
DFHMEADK	CSECT	ME domain - DFHADxxxx message set	14	03
DFHMEAIC	CSECT	ME domain - DFHAIxxxx message set simplified Chinese version	14	03
DFHMEAIE	CSECT	ME domain - DFHAIxxxx message set	14	03
DFHMEAIK	CSECT (OCO)	ME domain - DFHAIxxxx message set	14	03
DFHMEAMC	CSECT	ME domain - DFHAMxxxx message set simplified Chinese version	14	03
DFHMEAME	CSECT	ME domain - DFHAMxxxx message set	14	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHMEAMK	CSECT (OCO)	ME domain - DFHAMxxxx message set	14	03
DFHMEAPC	CSECT	ME domain - DFHAPxxxx message set simplified Chinese version	14	03
DFHMEAPE	CSECT	ME domain - DFHAPxxxx message set	14	03
DFHMEAPK	CSECT (OCO)	ME domain - DFHAPxxxx message set	14	03
DFHMEAUE	CSECT	ME domain - DFHAUxxxx message set	14	-
DFHMEBAC	CSECT	ME domain - DFHBAxxxx message set simplified Chinese version	14	03
DFHMEBAE	CSECT	ME domain - DFHBAxxxx message set	14	03
DFHMEBAK	CSECT (OCO)	ME domain - DFHBAxxxx message set	14	03
DFHMEBM	CSECT (OCO)	ME domain - batch message program	-	03
DFHMEBMA	DSECT	MEBM parameter list	OS	-
DFHMEBMM	Macro	MEBM request	OS	-
DFHMEBMT	CSECT (OCO)	MEBM trace interpretation data	-	03
DFHMEBRC	CSECT (OCO)	ME domain	14	03
DFHMEBRE	CSECT (OCO)	ME domain	14	03
DFHMEBRK	CSECT (OCO)	ME domain	14	03
DFHMEBU	CSECT (OCO)	ME domain - build message	-	03
DFHMEBUA	DSECT	MEBU parameter list	OS	-
DFHMEBUM	Macro	MEBU request	OS	-
DFHMEBUT	CSECT (OCO)	MEBU trace interpretation data	-	03
DFHMECAC	CSECT	ME domain - message set for GC/LC domains simplified Chinese version	14	03
DFHMECAE	CSECT	ME domain - DFHCAxxxx message set	14	03
DFHMECAK	CSECT	ME domain - DFHCAxxxx message set Japanese (Kanji) version	14	03
DFHMECCC	CSECT	ME domain - DFHCCxxxx message set simplified Chinese version	14	03
DFHMECCE	CSECT	ME domain - message set for GC/LC domains	14	03
DFHMECCK	CSECT (OCO)	ME domain - message set for GC/LC domains	14	03
DFHMECEC	CSECT	ME domain - DFHCExxxx message set simplified Chinese version	14	03
DFHMECEE	CSECT	ME domain - DFHCExxxx message set	14	03
DFHMECEK	CSECT (OCO)	ME domain - DFHCExxxx message set	14	03
DFHMECFE	CSECT	ME domain - DFHCFxxxx message set	14	-
DFHMECPC	CSECT	ME domain - DFHCPxxxx message set simplified Chinese version	14	03
DFHMECPE	CSECT	ME domain - DFHCPxxxx message set	14	03
DFHMECPK	CSECT (OCO)	ME domain - DFHCPxxxx message set	14	03
DFHMECRC	CSECT	ME domain - DFHCRxxxx message set simplified Chinese version	14	03
DFHMECRE	CSECT	ME domain - DFHCRxxxx message set	14	03
DFHMECRK	CSECT (OCO)	ME domain - DFHCRxxxx message set	14	03
DFHMECZC	CSECT	ME domain - DFHCZxxxx message set simplified Chinese version	14	03
DFHMECZE	CSECT	ME domain - DFHCZxxxx message set	14	03
DFHMECZK	CSECT (OCO)	ME domain - DFHCZxxxx message set	14	03
DFHMEDBC	CSECT	ME domain - DFHDBxxxx message set simplified Chinese version	14	03
DFHMEDBE	CSECT	ME domain - DFHDBxxxx message set	14	03
DFHMEDBK	CSECT (OCO)	ME domain - DFHDBxxxx message set	14	03
DFHMEDDC	CSECT	ME domain - message set for DD domain simplified Chinese version	14	03
DFHMEDDE	CSECT	ME domain - message set for DD domain	14	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library
DFHMEDDK	CSECT	ME domain - message set for DD domain	14 03
DFHMEDHC	CSECT	ME domain - message set for DH domain simplified Chinese version	14 03
DFHMEDHE	CSECT	ME domain - message set for DH domain	14 03
DFHMEDHK	CSECT	ME domain - message set for DH domain	14 03
DFHMEDM	CSECT (OCO)	ME domain - initialization/termination	- 03
DFHMEDMC	CSECT	ME domain - message set for DM domain simplified Chinese version	14 03
DFHMEDME	CSECT	ME domain - message set for DM domain	14 03
DFHMEDMK	CSECT	ME domain - message set for DM domain	14 03
DFHMEDSC	CSECT	ME domain - message set for DS domain simplified Chinese version	14 03
DFHMEDSE	CSECT	ME domain - message set for DS domain	14 03
DFHMEDSK	CSECT	ME domain - message set for DS domain	14 03
DFHMEDUC	CSECT	ME domain - message set for DU domain simplified Chinese version	14 03
DFHMEDUE	CSECT	ME domain - message set for DU domain	14 03
DFHMEDUF	CSECT (OCO)	SDUMP formatter for ME domain	- 03
DFHMEDUK	CSECT (OCO)	ME domain - message set for DU domain	14 03
DFHMEDXC	CSECT	ME domain - DFHDXxxxx message set simplified Chinese version	14 03
DFHMEDXE	CSECT	ME domain - DFHDXxxxx message set	14 03
DFHMEDXK	CSECT (OCO)	ME domain - DFHDXxxxx message set	14 03
DFHMEEJC	CSECT	ME domain - DFHEJxxxx message set simplified Chinese version	14 03
DFHMEEJE	CSECT	ME domain - DFHEJxxxx message set	14 03
DFHMEEJK	CSECT	ME domain - DFHEJxxxx message set	14 03
DFHMEEMC	CSECT	ME domain - DFHEMxxxx message set simplified Chinese version	14 03
DFHMEEME	CSECT	ME domain - DFHEMxxxx message set	14 03
DFHMEEMK	CSECT	ME domain - DFHEMxxxx message set	14 03
DFHMEERC	CSECT	ME domain - DFHERxxxx message set simplified Chinese version	14 03
DFHMEERE	CSECT	ME domain - DFHERxxxx message set	14 03
DFHMEERK	CSECT	ME domain - DFHERxxxx message set	14 03
DFHMEEXE	CSECT	ME domain - DFHEXxxxx message set	14 03
DFHMEFAC	CSECT	ME domain - DFHFAxxxx message set simplified Chinese version	14 03
DFHMEFAE	CSECT	ME domain - DFHFAxxxx message set	14 03
DFHMEFAK	CSECT (OCO)	ME domain - DFHFAxxxx message set	14 03
DFHMEFBC	CSECT	ME domain - DFHFBxxxx message set simplified Chinese version	14 03
DFHMEFBE	CSECT	ME domain - DFHFBxxxx message set	14 03
DFHMEFBK	CSECT (OCO)	ME domain - DFHFBxxxx message set	14 03
DFHMEFCC	CSECT	ME domain - DFHFCxxxx message set simplified Chinese version	14 03
DFHMEFCE	CSECT	ME domain - DFHFCxxxx message set	14 03
DFHMEFCK	CSECT (OCO)	ME domain - DFHFCxxxx message set	14 03
DFHMEFDC	CSECT	ME domain - DFHFDxxxx message set simplified Chinese version	14 03
DFHMEFDE	CSECT	ME domain - DFHFDxxxx message set	14 03
DFHMEFDK	CSECT (OCO)	ME domain - DFHFDxxxx message set	14 03
DFHMEFEC	CSECT	ME domain - DFHFExxxx message set simplified Chinese version	14 03
DFHMEFEE	CSECT	ME domain - DFHFExxxx message set	14 03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHMEFEK	CSECT (OCO)	ME domain - DFHFExxxx message set	14	03
DFHMEFO	CSECT (OCO)	ME domain - format message subroutine	-	03
DFHMEFOA	DSECT	MEFO parameter list	OS	-
DFHMEFOM	Macro	MEFO request	OS	-
DFHMEFOT	CSECT (OCO)	MEFO trace interpretation data	-	03
DFHMEICC	CSECT	ME domain - DFHICxxxx message set simplified Chinese version	14	03
DFHMEICE	CSECT	ME domain - DFHICxxxx message set	14	03
DFHMEICK	CSECT (OCO)	ME domain - DFHICxxxx message set	14	03
DFHMEIEC	CSECT	ME domain - DFHIExxxx message set simplified Chinese version	14	-
DFHMEIEE	CSECT	ME domain - DFHIExxxx message set	14	-
DFHMEIEK	CSECT (OCO)	ME domain - DFHIExxxx message set	14	-
DFHMEIIC	CSECT	ME domain - DFHIIxxxx message set simplified Chinese version	14	03
DFHMEIIE	CSECT	ME domain - DFHIIxxxx message set	14	03
DFHMEIIK	CSECT (OCO)	ME domain - DFHIIxxxx message set	14	03
DFHMEIN	CSECT (OCO)	ME domain - inquire message data	-	03
DFHMEINA	DSECT	MEIN parameter list	OS	-
DFHMEINC	DSECT	ME domain - DFHINxxxx message set simplified Chinese version	14	03
DFHMEINE	DSECT	ME domain - DFHINxxxx message set	14	03
DFHMEINK	DSECT	ME domain - DFHINxxxx message set Japanese (Kanji) version	14	03
DFHMEINM	Macro	MEIN request	OS	-
DFHMEINT	CSECT (OCO)	MEIN trace interpretation data	-	03
DFHMEIRC	CSECT	ME domain - DFHIRxxxx message set simplified Chinese version	14	03
DFHMEIRE	CSECT	ME domain - DFHIRxxxx message set	14	03
DFHMEIRK	CSECT (OCO)	ME domain - DFHIRxxxx message set Japanese (Kanji) version	14	03
DFHMEJCC	CSECT	ME domain - DFHJCxxxx message set simplified Chinese version	14	03
DFHMEJCE	CSECT	ME domain - DFHJCxxxx message set	14	03
DFHMEJCK	CSECT (OCO)	ME domain - DFHJCxxxx message set	14	03
DFHMEKCC	CSECT	ME domain - DFHKCxxxx message set simplified chinese version	14	03
DFHMEKCE	CSECT	ME domain - DFHKCxxxx message set	14	03
DFHMEKCK	CSECT (OCO)	ME domain - DFHKCxxxx message set Japanese (Kanji) version	14	03
DFHMEKEC	CSECT	ME domain - DFHKExxxx message set simplified chinese version	14	03
DFHMEKEE	CSECT	ME domain - message set for KE domain	14	03
DFHMEKEK	CSECT	ME domain - message set for KE domain	14	03
DFHMELDC	CSECT	ME domain - DFHLDxxxx message set simplified chinese version	14	03
DFHMELDE	CSECT	ME domain - message set for LD domain	14	03
DFHMELDK	CSECT	ME domain - message set for LD domain	14	03
DFHMELGC	CSECT	ME domain - DFHLGxxxx message set simplified Chinese version	14	03
DFHMELGE	CSECT	ME domain - DFHLGxxxx message set	14	03
DFHMELGK	CSECT	ME domain - DFHLGxxxx message set Japanese (Kanji) version	14	03
DFHMELMC	CSECT	ME domain - DFHLMxxxx message set simplified Chinese version	14	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library
DFHMELME	CSECT	ME domain - message set for LM domain	14 03
DFHMELMK	CSECT	ME domain - DFHLMxxxx message set Japanese (Kanji) version	14 03
DFHMEMCC	CSECT	ME domain - DFHMCxxxx message set simplified Chinese version	14 03
DFHMEMCE	CSECT	ME domain - DFHMCxxxx message set	14 03
DFHMEMCK	CSECT (OCO)	ME domain - DFHMCxxxx message set Japanese (Kanji) version	14 03
DFHMEME	CSECT (OCO)	ME domain - main functions	- 03
DFHMEMEC	CSECT	ME domain - DFHMExxxx message set simplified Chinese version	14 03
DFHMEMEA	DSECT	MEME parameter list	OS -
DFHMEMEE	CSECT	ME domain - DFHMExxxx message set	14 03
DFHMEMEK	CSECT (OCO)	ME domain - main functions	14 03
DFHMEMEM	Macro	MEME request	OS -
DFHMEMET	CSECT (OCO)	MEME trace interpretation data	- 03
DFHMEMNC	CSECT	ME domain - DFHMNxxxx message set simplified Chinese version	14 03
DFHMEMNE	CSECT	ME domain - message set for MN domain	14 03
DFHMEMNK	CSECT	ME domain - message set for MN domain	14 03
DFHMEMUC	CSECT	ME domain - DFHMUxxxx message set simplified Chinese version	14 03
DFHMEMUE	CSECT	ME domain - DFHMUxxxx message set	14 03
DFHMEMUK	CSECT	ME domain - DFHMUxxxx message set	14 03
DFHMENCE	CSECT	ME domain - DFHNCxxxx message set	14 -
DFHMENQC	CSECT	ME domain - DFHMQxxxx message set simplified Chinese version	14 03
DFHMENQE	CSECT	ME domain - DFHNQxxxx message set	14 03
DFHMENQK	CSECT	ME domain - DFHNQxxxx message set	14 03
DFHMEOTC	CSECT	ME domain - DFHOTxxxx message set simplified Chinese version	14 03
DFHMEOTE	CSECT	ME domain - DFHOTxxxx message set	14 03
DFHMEOTK	CSECT	ME domain - DFHOTxxxx message set	14 03
DFHMEPAC	CSECT	ME domain - DFHPAxxxx message set simplified Chinese version	14 03
DFHMEPAE	CSECT	ME domain - message set for PA domain	14 03
DFHMEPAK	CSECT	ME domain - message set for PA domain	14 03
DFHMEPCC	CSECT	ME domain - DFHPCxxxx message set simplified Chinese version	14 03
DFHMEPCE	CSECT	ME domain - DFHPCxxxx message set	14 03
DFHMEPCK	CSECT (OCO)	ME domain - DFHPCxxxx message set Japanese (Kanji) version	14 03
DFHMEPGC	CSECT	ME domain - DFHPGxxxx message set simplified Chinese version	14 03
DFHMEPGE	CSECT	ME domain - DFHPGxxxx message set	14 03
DFHMEPGK	CSECT (OCO)	ME domain - DFHPGxxxx message set Japanese (Kanji) version	14 03
DFHMEPRC	CSECT	ME domain - DFHPRxxxx message set simplified Chinese version	14 03
DFHMEPRE	CSECT	ME domain - DFHPRxxxx message set	14 03
DFHMEPRK	CSECT (OCO)	ME domain - DFHPRxxxx message set Japanese (Kanji) version	14 03
DFHMEPSC	CSECT	ME domain - DFHPSxxxx message set simplified Chinese version	14 03
DFHMEPSE	CSECT	ME domain - DFHPSxxxx message set	14 03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library
DFHMEPSK	CSECT (OCO)	ME domain - DFHPSxxxx message set	14 03
DFHMERDC	CSECT	ME domain - DFHRDxxxx message set simplified Chinese version	14 03
DFHMERDE	CSECT	ME domain - DFHRDxxxx message set	14 03
DFHMERDK	CSECT (OCO)	ME domain - DFHRDxxxx message set Japanese (Kanji) version	14 03
DFHMERMC	CSECT	ME domain - DFHRMxxxx message set simplified Chinese version	14 03
DFHMERME	CSECT	ME domain - DFHRMxxxx message set	14 03
DFHMERMK	CSECT (OCO)	ME domain - DFHRMxxxx message set Japanese (Kanji) version	14 03
DFHMEROC	CSECT	ME domain - DFHRPxxxx message set simplified Chinese version	14 03
DFHMEROE	CSECT	ME domain - DFHRPxxxx message set	14 03
DFHMEROK	CSECT (OCO)	ME domain - DFHRPxxxx message set Japanese (Kanji) version	14 03
DFHMERPC	CSECT	ME domain - DFHRPxxxx message set simplified Chinese version	14 03
DFHMERPE	CSECT	ME domain - DFHRPxxxx message set	14 03
DFHMERPK	CSECT (OCO)	ME domain - DFHRPxxxx message set Japanese (Kanji) version	14 03
DFHMERQC	CSECT	ME domain - DFHRPxxxx message set simplified Chinese version	14 03
DFHMERQE	CSECT	ME domain - DFHRPxxxx message set	14 03
DFHMERQK	CSECT (OCO)	ME domain - DFHRPxxxx message set Japanese (Kanji) version	14 03
DFHMERRC	CSECT	ME domain - DFHRPxxxx message set simplified Chinese version	14 03
DFHMERRE	CSECT	ME domain - DFHRPxxxx message set	14 03
DFHMERRK	CSECT (OCO)	ME domain - DFHRPxxxx message set Japanese (Kanji) version	14 03
DFHMERSC	CSECT	ME domain - DFHRSxxxx message set simplified Chinese version	14 03
DFHMERSE	CSECT	ME domain - DFHRSxxxx message set	14 03
DFHMERSK	CSECT (OCO)	ME domain - DFHRSxxxx message set Japanese (Kanji) version	14 03
DFHMERTC	CSECT	ME domain - DFHRTxxxx message set simplified Chinese version	14 03
DFHMERTE	CSECT	ME domain - DFHRTxxxx message set	14 03
DFHMERTK	CSECT (OCO)	ME domain - DFHRTxxxx message set Japanese (Kanji) version	14 03
DFHMERUC	CSECT	ME domain - DFHRUxxxx message set simplified Chinese version	14 03
DFHMERUE	CSECT	ME domain - DFHRUxxxx message set	14 03
DFHMERUK	CSECT	ME domain - DFHRUxxxx message set	14 03
DFHMERXC	CSECT	ME domain - DFHRXxxxx message set simplified Chinese version	14 03
DFHMERXE	CSECT	ME domain - DFHRXxxxx message set	14 03
DFHMERXK	CSECT	ME domain - DFHRXxxxx message set	14 03
DFHMERZC	CSECT	ME domain - DFHRZxxxx message set simplified Chinese version	14 03
DFHMERZE	CSECT	ME domain - DFHRZxxxx message set	14 03
DFHMERZK	CSECT	ME domain - DFHRZxxxx message set	14 03
DFHMESHC	CSECT	ME domain - DFHSHxxxx message set simplified Chinese version	14 03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHMESHE	CSECT	ME domain - DFHSHxxxx message set	14	03
DFHMESHK	CSECT	ME domain - DFHSHxxxx message set	14	03
DFHMESIC	CSECT	ME domain - DFHSIxxxx message set simplified Chinese version	14	03
DFHMESIE	CSECT	ME domain - DFHSIxxxx message set	14	03
DFHMESIK	CSECT (OCO)	ME domain - DFHSIxxxx message set Japanese (Kanji) version	14	03
DFHMESJC	CSECT	ME domain - DFHSJxxxx message set simplified Chinese version	14	03
DFHMESJE	CSECT	ME domain - DFHSJxxxx message set	14	03
DFHMESJK	CSECT (OCO)	ME domain - DFHSJxxxx message set Japanese (Kanji) version	14	03
DFHMESKC	CSECT	ME domain - DFHSKxxxx message set simplified Chinese version	14	03
DFHMESKE	CSECT	ME domain - DFHSKxxxx message set	14	03
DFHMESKK	CSECT	ME domain - DFHSKxxxx message set	14	03
DFHMESMC	CSECT	ME domain - DFHSMxxxx message set simplified Chinese version	14	03
DFHMESME	CSECT	ME domain - message set for SM domain	14	03
DFHMESMK	CSECT	ME domain - message set for SM domain	14	03
DFHMESNC	CSECT	ME domain - DFHSNxxxx message set simplified Chinese version	14	03
DFHMESNE	CSECT	ME domain - DFHSNxxxx message set	14	03
DFHMESNK	CSECT (OCO)	ME domain - DFHSNxxxx message set Japanese (Kanji) version	14	03
DFHMESOC	CSECT	ME domain - DFHSOxxxx message set simplified Chinese version	14	03
DFHMESOE	CSECT	ME domain - DFHSOxxxx message set	14	03
DFHMESOK	CSECT (OCO)	ME domain - DFHSOxxxx message set Japanese (Kanji) version	14	03
DFHMESR	CSECT (OCO)	ME domain - SIT overrides collection	-	03
DFHMESRA	DSECT	MESR parameter list	OS	-
DFHMESRC	CSECT	ME domain - DFHSRxxxx message set simplified Chinese version	14	03
DFHMESRE	CSECT	ME domain - DFHSRxxxx message set	14	03
DFHMESRK	CSECT	ME domain - DFHSRxxxx message set	14	03
DFHMESRM	Macro	MESR request	OS	-
DFHMESRT	CSECT (OCO)	MESR trace interpretation data	-	03
DFHMESTC	CSECT	ME domain - message set for ST domain simplified Chinese version	14	03
DFHMESTE	CSECT	ME domain - message set for ST domain	14	03
DFHMESTK	CSECT (OCO)	ME domain - message set for ST domain Japanese (Kanji) version	14	03
DFHMESZC	CSECT (OCO)	ME domain - DFHSZxxxx message set (FEPI) simplified Chinese version	14	03
DFHMESZE	CSECT (OCO)	ME domain - DFHSZxxxx message set (FEPI)	14	03
DFHMESZK	CSECT (OCO)	ME domain - DFHSZxxxx message set (FEPI) Japanese (Kanji) version	14	03
DFHMETCC	CSECT	ME domain - DFHTCxxxx message set simplified Chinese version	14	03
DFHMETCE	CSECT	ME domain - DFHTCxxxx message set	14	03
DFHMETCK	CSECT (OCO)	ME domain - DFHTCxxxx message set Japanese (Kanji) version	14	03
DFHMETDC	CSECT	ME domain - DFHTDxxxx message set simplified Chinese version	14	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHMETDE	CSECT	ME domain - DFHTDxxxx message set	14	03
DFHMETDK	CSECT	ME domain - DFHTDxxxx message set Japanese (Kanji) version	14	03
DFHMETFC	CSECT	ME domain - DFHTFxxxx message set simplified Chinese version	14	03
DFHMETFE	CSECT	ME domain - DFHTFxxxx message set	14	03
DFHMETFK	CSECT (OCO)	ME domain - DFHTFxxxx message set Japanese (Kanji) version	14	03
DFHMETIC	CSECT	ME domain - DFHTIxxxx message set simplified Chinese version	14	03
DFHMETIE	CSECT	ME domain - message set for TI domain	14	03
DFHMETIK	CSECT	ME domain - message set for TI domain	14	03
DFHMETMC	CSECT	ME domain - DFHTMxxxx message set simplified Chinese version	14	03
DFHMETME	CSECT	ME domain - DFHTMxxxx message set	14	03
DFHMETMK	CSECT	ME domain - DFHTMxxxx message set Japanese (Kanji) version	14	03
DFHMETOC	CSECT	ME domain - DFHTOxxxx message set simplified Chinese version	14	03
DFHMETOE	CSECT	ME domain - DFHTOxxxx message set	14	03
DFHMETOK	CSECT (OCO)	ME domain - DFHTOxxxx message set Japanese (Kanji) version	14	03
DFHMETPC	CSECT	ME domain - DFHTPxxxx message set simplified Chinese version	14	03
DFHMETPE	CSECT	ME domain - DFHTPxxxx message set	14	03
DFHMETPK	CSECT (OCO)	ME domain - DFHTPxxxx message set Japanese (Kanji) version	14	03
DFHMETRC	CSECT	ME domain - message set for TR domain simplified Chinese version	14	03
DFHMETRE	CSECT	ME domain - message set for TR domain	14	03
DFHMETRI	CSECT (OCO)	Trace interpreter for ME domain	-	03
DFHMETRK	CSECT (OCO)	ME domain - message set for TR domain Japanese (Kanji) version	14	03
DFHMETSC	CSECT	ME domain - DFHTSxxxx message set simplified Chinese version	14	03
DFHMETSE	CSECT	ME domain - DFHTSxxxx message set	14	03
DFHMETSK	CSECT (OCO)	ME domain - DFHTSxxxx message set Japanese (Kanji) version	14	03
DFHMET1	CSECT	ME domain - DFHMET1x online message table	14	03
DFHMET1E	CSECT	DFHMEU base messages link-edit module	14	-
DFHMET2	CSECT (OCO)	ME domain - DFHMET2x offline translator message table	-	03
DFHMET3	CSECT (OCO)	ME domain - DFHMET3x offline message table for DFHSTUP	-	03
DFHMET4	CSECT (OCO)	Offline message table for EXCI	-	03
DFHMET5	CSECT	ME domain - DFHMET5x online message table	OS	03
DFHMET6	CSECT	ME domain - DFHMET6x online message table	-	03
DFHMET5E	CSECT	DFHMEU ONC RPS messages link-edit module	14	-
DFHMET9	CSECT	ME domain - DFHMET9x online message table	OS	03
DFHMEU	CSECT	Message translation utility program	-	03
DFHMEUA	DSECT (OCO)	Message editing utility parameter list	-	03
DFHMEUC	CSECT (OCO)	Message editing utility copy message dataset	-	03
DFHMEUCL	CSECT (OCO)	Message editing utility copy message dataset	06	-
DFHMEUD	CSECT (OCO)	Message editing utility set/validate system defaults	-	03
DFHMEUE	CSECT (OCO)	Message editing utility edit message	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHMEUL	CSECT (OCO)	Message editing utility compile, assemble and link-edit message data sets	-	03
DFHMEULT	CSECT (OCO)	Message editing utility CLIST to create language codes table	06	-
DFHMEUM	Macro (OCO)	Message editing utility ISPF editor profile	-	03
DFHMEUP	CSECT (OCO)	Message editing utility display PTF panel and submit PTF job	-	03
DFHMEUPC	CSECT	ME domain - message set for UP domain simplified Chinese version	14	03
DFHMEUPE	CSECT (OCO)	ME domain - DFHUPxxxx message set	14	03
DFHMEUPK	CSECT (OCO)	ME domain - DFHUPxxxx message set	14	03
DFHMEUSC	CSECT (OCO)	Message editing utility check state of message data set simplified Chinese version	14	03
DFHMEUSE	CSECT (OCO)	Message editing utility check state of message data set	14	03
DFHMEUSK	CSECT (OCO)	Message editing utility check state of message data set Japanese (Kanji) version	14	03
DFHMEUU	CSECT (OCO)	Message editing utility compare PTF and English message data sets	-	03
DFHMEU00	CSECT	Message editing utility help index panel	16	-
DFHMEU01	CSECT	Message editing utility main help panel 1	16	-
DFHMEU10	CSECT	Message editing utility main panel	16	-
DFHMEU11	CSECT	Message editing utility main help panel 2	16	-
DFHMEU12	CSECT	Message editing utility main help panel 3	16	-
DFHMEU20	CSECT	Message editing utility set defaults panel (part 1 of 2)	16	-
DFHMEU21	CSECT	Message editing utility set defaults (part 1) help panel 1	16	-
DFHMEU22	CSECT	Message editing utility set defaults (part 1) help panel 2	16	-
DFHMEU30	CSECT	Message editing utility set defaults panel (part 2 of 2)	16	-
DFHMEU31	CSECT	Message editing utility set defaults (part 2) help panel	16	-
DFHMEU40	CSECT	Message editing utility language selection panel	16	-
DFHMEU41	CSECT	Message editing utility language selection help panel	16	-
DFHMEU50	CSECT	Message editing utility message selection panel	16	-
DFHMEU51	CSECT	Message editing utility message selection help panel	16	-
DFHMEU60	CSECT	Message editing utility message edit panel	16	-
DFHMEU61	CSECT	Message editing utility message edit help panel	16	-
DFHMEU70	CSECT	Message editing utility apply PTF updates panel	16	-
DFHMEU71	CSECT	Message editing utility apply PTF updates help panel	16	-
DFHMEWBC	CSECT (OCO)	ME domain	14	03
DFHMEWBE	CSECT (OCO)	ME domain	14	03
DFHMEWBK	CSECT (OCO)	ME domain	14	03
DFHMEWS	CSECT (OCO)	ME domain - write symptom string to SYS1.LOGREC	-	03
DFHMEWSA	DSECT	MEWS parameter list	OS	-
DFHMEWSM	Macro	MEWS request	OS	-
DFHMEWST	CSECT (OCO)	MEWS trace interpretation data	-	03
DFHMEWT	CSECT (OCO)	ME domain - WTOR service routine	-	03
DFHMEWTA	DSECT	MEWT parameter list	OS	-
DFHMEWTM	Macro	MEWT request	OS	-
DFHMEWTT	CSECT (OCO)	MEWT trace interpretation data	-	03
DFHMEXAC	CSECT	ME domain - message set for XA domain simplified Chinese version	14	03
DFHMEXAE	CSECT	ME domain - DFHXAxxxx message set	14	03
DFHMEXAK	CSECT	ME domain - DFHXAxxxx message set	14	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library
DFHMEXCC	CSECT	ME domain - message set for XC domain simplified Chinese version	14 03
DFHMEXCE	CSECT	ME domain - DFHXCxxxx message set	14 03
DFHMEXCK	CSECT	ME domain - DFHXCxxxx message set	14 03
DFHMEXGC	CSECT	ME domain - DFHXGxxxx message set simplified Chinese version	14 03
DFHMEXGE	CSECT	ME domain - DFHXGxxxx message set	14 03
DFHMEXGK	CSECT (OCO)	ME domain - DFHXGxxxx message set Japanese (Kanji) version	14 03
DFHMEXMC	CSECT	ME domain - DFHXMxxxx message set simplified Chinese version	14 03
DFHMEXME	CSECT	ME domain - DFHXMxxxx message set	14 03
DFHMEXMK	CSECT (OCO)	ME domain - DFHXMxxxx message set Japanese (Kanji) version	14 03
DFHMEXOC	CSECT	ME domain - DFHXOxxxx message set simplified Chinese version	14 03
DFHMEXOE	CSECT	ME domain - DFHXOxxxx message set	14 03
DFHMEXOK	CSECT	ME domain - DFHXOxxxx message set	14 03
DFHMEXQE	CSECT	ME domain - DFHXQxxxx message set	14 -
DFHMEXSC	CSECT	ME domain - DFHXSxxxx message set simplified Chinese version	14 03
DFHMEXSE	CSECT	ME domain - DFHXSxxxx message set	14 03
DFHMEXSK	CSECT (OCO)	ME domain - DFHXSxxxx message set Japanese (Kanji) version	14 03
DFHMEZAC	CSECT	ME domain - DFHZAxxxx message set simplified Chinese version	14 03
DFHMEZAE	CSECT	ME domain - DFHZAxxxx message set	14 03
DFHMEZAK	CSECT (OCO)	ME domain - DFHZAxxxx message set Japanese (Kanji) version	14 03
DFHMEZBC	CSECT	ME domain - DFHZBxxxx message set simplified Chinese version	14 03
DFHMEZBE	CSECT	ME domain - DFHZBxxxx message set	14 03
DFHMEZBK	CSECT (OCO)	ME domain - DFHZBxxxx message set Japanese (Kanji) version	14 03
DFHMEZCC	CSECT	ME domain - DFHZCxxxx message set simplified Chinese version	14 03
DFHMEZCE	CSECT	ME domain - DFHZCxxxx message set	14 03
DFHMEZCK	CSECT (OCO)	ME domain - DFHZCxxxx message set Japanese (Kanji) version	14 03
DFHMEZDC	CSECT	ME domain - DFHZDxxxx message set simplified Chinese version	14 03
DFHMEZDE	CSECT	ME domain - DFHZDxxxx message set	14 03
DFHMEZDK	CSECT (OCO)	ME domain - DFHZDxxxx message set Japanese (Kanji) version	14 03
DFHMEZEC	CSECT	ME domain - DFHZExxxx message set simplified Chinese version	14 03
DFHMEZEE	CSECT	ME domain - DFHZExxxx message set	14 03
DFHMEZEK	CSECT (OCO)	ME domain - DFHZExxxx message set Japanese (Kanji) version	14 03
DFHMEZNC	CSECT	ME domain - DFHZNxxxx message set simplified Chinese version	14 03
DFHMEZNE	CSECT	ME domain - DFHZNxxxx message set	14 03
DFHMEZNK	CSECT (OCO)	ME domain - DFHZNxxxx message set Japanese (Kanji) version	14 03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHME00C	CSECT	ME domain - NLS message language globals simplified Chinese version	14	03
DFHME00E	CSECT	ME domain - NLS message language globals	14	03
DFHME00K	CSECT (OCO)	ME domain - NLS message language globals Japanese (Kanji) version	14	03
DFHME01E	CSECT	ME domain - NLS message language globals	14	-
DFHME1UC	CSECT	ME domain	14	03
DFHME1UE	CSECT	ME domain - DFH1Uxx message set	14	03
DFHME1UK	CSECT (OCO)	ME domain - DFH1Uxx message set	14	03
DFHME42E	CSECT	ME domain - DFH42xx message set	14	-
DFHME70C	CSECT	ME domain - DFH70xx message set simplified Chinese version	14	03
DFHME70E	CSECT	ME domain - DFH70xx message set	14	03
DFHME70K	CSECT (OCO)	ME domain - DFH70xx message set	14	03
DFHME71C	CSECT	ME domain - DFH71xx message set simplified Chinese version	14	03
DFHME71E	CSECT	ME domain - DFH71xx message set	14	03
DFHME71K	CSECT (OCO)	ME domain - DFH71xx message set	14	03
DFHME72C	CSECT	ME domain - DFH72xx message set simplified Chinese version	14	03
DFHME72E	CSECT	ME domain - DFH72xx message set	14	03
DFHME72K	CSECT (OCO)	ME domain - DFH72xx message set	14	03
DFHMGM	Macro	Message prototype macro	11	-
DFHMGMI0	Macro	Message prototype literal macro-1	11	-
DFHMGMI1	Macro	Message prototype literal macro-2	11	-
DFHMGPME	CSECT	DFHMGP NLS message support	OS	03
DFHMGP00	CSECT	DFHMGP error message find	OS	03
DFHMGT	CSECT	Message generation table	11	03
DFHMGT01	CSECT	Subsystem interface message table segment	11	-
DFHMGT20	CSECT	Message generation table segment	11	-
DFHMGT21	CSECT	Message generation table segment	11	-
DFHMGT22	CSECT	Message generation table segment	11	-
DFHMGT24	CSECT	Message generation table segment	11	-
DFHMGT26	CSECT	Message generation table segment	11	-
DFHMGT33	CSECT	Message generation table segment	11	-
DFHMGT34	CSECT	Message generation table segment	11	-
DFHMGT35	CSECT	Message generation table segment	11	-
DFHMGT37	CSECT	Message generation table segment	11	-
DFHMGT44	CSECT	Message generation table segment	11	-
DFHMGT49	CSECT	Message generation table segment	11	-
DFHMGT50	CSECT	Message generation table segment	11	-
DFHMGT85	CSECT	Message generation table segment	11	-
DFHMGT90	CSECT	Message generation table segment	11	-
DFHMIN	Source	BMS 3270 input mapping	OS	-
DFHMIRS	CSECT	ISC request shipping - mirror program	OS	03
DFHMKDIR	Other		-	02
DFHMKEYS	CSECT	Alias for MEUKEYS	16	-
DFHML1	CSECT	BMS LU1 printer mapping program	OS	03
DFHMN	Macro	MN domain - inline request	OS	-
DFHMNDEF	Macro	MN domain - some control blocks	OS	-
DFHMNDM	CSECT (OCO)	MN domain - initialization/termination	-	03
DFHMNDUF	CSECT (OCO)	SDUMP formatter for MN domain	-	03
DFHMNDUP	CSECT (OCO)	Monitoring dictionary utility	-	03
DFHMNEXC	Macro	MN domain - monitoring exception record	11	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHMNGDS	DSECT	MN domain - global statistics	11	-
DFHMNGDS	DSECT	MN domain - global statistics	C2	07
DFHMNMN	CSECT (OCO)	MN domain - functions	-	03
DFHMNMNA	DSECT	MNMN parameter list	OS	-
DFHMNMNM	Macro	MNMN request	OS	-
DFHMNMNT	CSECT	MNMN trace interpretation data	OS	03
DFHMNMNX	Macro	MNMN request (XPI)	11	-
DFHMNMNY	DSECT	MNMN parameter list (XPI)	11	-
DFHMNNT	CSECT (OCO)	MN domain - XM notify gate	-	03
DFHMNPBI	Macro	MN domain - access to MVS WLM performance block token	OS	-
DFHMNPDA	CSECT	Monitoring facility performance class record	19	-
DFHMNSMF	Macro	MN domain - monitoring SMF header and SMF product section	11	-
DFHMNSR	CSECT (OCO)	MN domain - services	-	03
DFHMNSRA	DSECT	MNSR parameter list	OS	-
DFHMNSRM	Macro	MNSR request	OS	-
DFHMNSRT	CSECT	MNSR trace interpretation data	OS	03
DFHMNST	CSECT (OCO)	MN domain - statistics services	-	03
DFHMNSU	CSECT (OCO)	MN domain - subroutines	-	03
DFHMNSUA	DSECT	MNSU parameter list	OS	-
DFHMNSUM	Macro	MNSU request	OS	-
DFHMNSUT	CSECT	MNSU trace interpretation data	OS	03
DFHMNSVC	CSECT (OCO)	MN domain - authorized service routine	-	03
DFHMNTDS	DSECT	MN domain - transaction monitoring data	11	-
DFHMNTDS	DSECT	MN domain - transaction monitoring data	C2	07
DFHMNTI	CSECT (OCO)	MN domain - timer gate	-	03
DFHMNTRI	CSECT (OCO)	Trace interpreter for MN domain	-	03
DFHMNUE	CSECT (OCO)	MN domain - user exit service	-	03
DFHMNXM	CSECT (OCO)	MN domain functional gate	-	03
DFHMNXMT	DSECT	MNXM translate tables	-	03
DFHMOVE	Macro	Domain call argument MOVE macro	OS	-
DFHMPARS	CSECT	Parameter syntax checking	OS	-
DFHMRCDS	DSECT	Transient data VSAM control	OS	-
DFHMRDUF	CSECT (OCO)	MRO SDUMP formatter	-	03
DFHMROQM	Macro	MRO work queue manager interface	OS	-
DFHMROQP	CSECT	MRO work queue manager - enable/disable	OS	03
DFHMROSM	Macro	MRO work queue manager quickcell interface	OS	-
DFHMRQDS	DSECT	MRO work queue manager control blocks	OS	-
DFHMRXM	CSECT	TF XM transaction attach	-	03
DFHMSCAN	CSECT	Macro scan utility	OS	03
DFHMSD	Macro	Generate BMS map set definition	11	-
DFHMSET	CSECT	Parameter syntax checking record	OS	-
DFHMSG	Macro	Generate a message	11	-
DFHMSGIF	CSECT	CZ Direct_to_CICS	-	03
DFHMSG00	CSECT	MEU MEU00x message set (alias MEU00)	12	-
DFHMSG01	CSECT	MEU MEU01x message set (alias MEU01)	12	-
DFHMSG02	CSECT	MEU MEU02x message set (alias MEU02)	12	-
DFHMSG03	CSECT	MEU MEU03x message set (alias MEU03)	12	-
DFHMSG04	CSECT	MEU MEU04x message set (alias MEU04)	12	-
DFHMSG05	CSECT	MEU MEU05x message set (alias MEU05)	12	-
DFHMSGEN	Macro	Generate messages in BMS modules	OS	-
DFHMSP	CSECT	Message switching program	OS	03
DFHMSPUT	Macro	Put messages to terminals in BMS	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHMSRCA	Symbolic	Magnetic slot reader control values	11	-
DFHMSRCA	Symbolic	Magnetic slot reader control values	C2	07
DFHMSRCA	Symbolic	Magnetic slot reader control values	D3	08
DFHMSX	Symbolic		11	-
DFHMVRMS	CSECT (OCO)	MVS recovery/termination manager RESMGR exit stub	-	03
DFHMWCDS	DSECT	Transient data wait control	OS	-
DFHMXP	CSECT	Local queuing shipper	OS	03
DFHM32	CSECT	BMS 3270 mapping	OS	-
DFHM32A$	CSECT	BMS 3270 mapping (standard)	OS	03
DFHM321$	CSECT	BMS 3270 mapping (full)	OS	03
DFHNCASM	Macro	Named counter service interface	11	-
DFHNCC	DSECT	Named counter service interface	-	08
DFHNCCF	DSECT	Named counter service interface	-	03
DFHNCCN	DSECT	Named counter service interface	-	03
DFHNCOB	DSECT	Named counter service interface	-	07
DFHNCDF	DSECT	Named counter server AXM definitions	-	03
DFHNCEN	DSECT	NC ENF event interface	-	03
DFHNCEQU	Macro	Named counter server interface	11	-
DFHNCIF	CSECT	Named counter server interface	-	03
DFHNCMN	CSECT	Named counter server main program	-	03
DFHNCMS	CSECT	Named counter server messages	-	03
DFHNCO	Macro	Named counter option table definition	11	-
DFHNCOP	CSECT	Named counter server operator commands	-	03
DFHNCOPT	CSECT	Named counter server sample option table	19	03
DFHNCPLI	CSECT	Named counter service interface	17	-
DFHNCPR	CSECT	Named counter server parameter routine	-	03
DFHNCPS	CSECT	Named counter server pool selection	-	03
DFHNCRL	CSECT	Named counter server pool reload	-	03
DFHNCRQ	CSECT	Named counter server request routine	-	03
DFHNCRS	CSECT	NC ARM Restart Support	-	03
DFHNCST	CSECT	Named counter server statistics support	-	03
DFHNCS4D	Macro	Named counter server list str stats	11	-
DFHNCS5D	Macro	Named counter server storage statistics	11	-
DFHNCTR	CSECT	Named counter server interface stub	-	03
DFHNCUL	CSECT	Named counter server pool unload	-	03
DFHNEPCA	DSECT	NEP communication area	D2	-
DFHNEPCA	Macro	NEP communication area	11	-
DFHNOTIT	CSECT		-	03
DFHNQDM	CSECT	NQ domain management	-	03
DFHNQDUF	CSECT	NQ offline dump formatting	-	03
DFHNQED	CSECT	NQED format enqueue/dequeue	-	03
DFHNQEDA	CSECT	NQED parameter list	OS	-
DFHNQEDM	Macro	NQED request	OS	-
DFHNQEDT	DSECT	NQED translate tables	-	03
DFHNQEDX	Macro		11	-
DFHNQEDY	Macro		11	-
DFHNQGDS	CSECT	NQ enqueue manager statistics	11	-
DFHNQGDS	CSECT	NQ enqueue manager statistics	C2	07
DFHNQIB	CSECT	NQ inquire/browse module	-	03
DFHNQIBA	CSECT	NQIB parameter list	OS	-
DFHNQIBM	Macro	NQIB request	OS	-
DFHNQIBT	DSECT	NQIB translate tables	-	03
DFHNQIE	CSECT	NQ default enqueue interpreter	-	03
DFHNQNQ	CSECT	NQ main functions	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHNQNQA	CSECT	NQNQ parameter list	OS	-
DFHNQNQM	Macro	NQNQ request	OS	-
DFHNQNQT	DSECT	NQNQ translate tables	-	03
DFHNQRN	CSECT	Sysplex resource names services	-	03
DFHNQRNA	Other	NQRN interface parameter area	OS	-
DFHNQRNM	Macro	DFHNQRN interface macro	OS	-
DFHNQRNT	CSECT		-	03
DFHNQST	CSECT (OCO)	NQ statistics	-	03
DFHNQTRI	CSECT (OCO)	NQ offline trace interpretation	-	03
DFHNQUED	Macro	EXEC arguement list for ENQ/DEQ user exits	11	-
DFHNXDUF	CSECT (OCO)	SDUMP control block index processor	-	03
DFHOPSRC	Other	JCL to install optional source tapes	02	-
DFHOSPWA	DSECT	BMS common control area	11	-
DFHOTCO	CSECT	OTCO CDURUN and Gate Module	-	03
DFHOTCOT	CSECT		-	03
DFHOTCPT	CSECT		-	03
DFHOTDM	CSECT	OT Domain Management	-	03
DFHOTDUF	CSECT	OT Domain Dump Formatting	-	03
DFHOTIS1	CSECT		-	03
DFHOTIS2	CSECT		-	03
DFHOTR	CSECT	OTS Resync Transaction	-	03
DFHOTRM	CSECT	Run Transaction Syncpoint Processor	-	03
DFHOTRP1	CSECT		-	03
DFHOTRS	CSECT	OTRS CDURUN and Gate Module	-	03
DFHOTRST	CSECT		-	03
DFHOTSU	CSECT	OTSU CDURUN and Gate Module	-	03
DFHOTSUT	CSECT		-	03
DFHOTTR	CSECT	OTTR CDURUN and Gate Module	-	03
DFHOTTRI	CSECT	OT Domain Trace Interpretation	-	03
DFHOTTRT	CSECT		-	03
DFHOTVP1	CSECT		-	03
DFHPADM	CSECT (OCO)	PA domain - initialization/termination	-	03
DFHPADUF	CSECT (OCO)	SDUMP formatter for PA domain	-	03
DFHPAGP	CSECT (OCO)	PA domain - get parameters service	-	03
DFHPAGPA	DSECT	PAGP parameter list	OS	-
DFHPAGPM	Macro	PAGP request	OS	-
DFHPAGPT	CSECT (OCO)	PAGP trace interpretation data	-	03
DFHPAIO	CSECT (OCO)	PA domain - communication with SYSIN data set and operator console	-	03
DFHPAIOA	DSECT	PAIO parameter list	OS	-
DFHPAIOM	Macro	PAIO request	OS	-
DFHPAIOT	CSECT (OCO)	PAIO trace interpretation data	-	03
DFHPAPL	Macro	DBCTL architected parameter list	OS	-
DFHPASY	CSECT (OCO)	PA domain - system initialization parameter checker and syntax analyzer	-	03
DFHPASYA	DSECT	PASY parameter list	OS	-
DFHPASYM	Macro	PASY request	OS	-
DFHPASYT	CSECT (OCO)	PASY trace interpretation data	-	03
DFHPATCH	Macro	Generate patch area	11	-
DFHPATRI	CSECT (OCO)	Trace interpreter for PA domain	-	03
DFHPBP	CSECT	BMS page and text build	OS	-
DFHPBPA$	CSECT	BMS page and text build (standard)	OS	03
DFHPBP1$	CSECT	BMS page and text build (full)	OS	03
DFHPC	Macro	Program service request	11	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHPCEDS	DSECT	EXEC argument list for Program Control	-	11
DFHPCEXT	CSECT	AP recovery point when called from kernel	OS	-
DFHPCOM	Macro	PEP communication area	11	-
DFHPCOMD	DSECT	PEP communication area	-	08
DFHPCPC2	CSECT	PCP interface to COBOL stub for OS/VS COBOL V1 R2.3 or R2.4 application programs	OS	03
DFHPCPG	CSECT	PM domain - interface program	-	03
DFHPCTDS	DSECT	Program control table	11	-
DFHPCTPF	Macro	Generate a profile entry	11	-
DFHPCUE	DSECT	Program control data block for user exits	11	-
DFHPCXDF	CSECT	DU domain - transaction dump formatter for program related areas	OS	03
DFHPDI	Macro	Generate BMS partition definition	11	-
DFHPDKW	CSECT (OCO)	SDUMP formatting - CICSDATA operand string validation	-	03
DFHPDX1	CSECT (OCO)	SDUMP formatting - control program	-	03
DFHPEP	CSECT	User-replaceable program error program	19	03
DFHPEPD	Sample	Program error program - C/370	-	19
DFHPESAD	Source	Program environment save area (PESA)	OS	-
DFHPGACD	Macro	PG domain - autoinstall exit program parameter list - Assembler	11	-
DFHPGACH	CSECT	PG domain - autoinstall exit program parameter list - C/370	-	08
DFHPGACL	CSECT	PG domain - autoinstall exit program parameter list - PL/I	P2	-
DFHPGACO	CSECT	PG domain - autoinstall exit program parameter list - COBOL	C2	-
DFHPGADS	DSECT	BMS page control area	OS	-
DFHPGADX	CSECT	Program autoinstall exit - Assembler	19	03
DFHPGAHX	Sample	Program autoinstall exit - C/370	-	19
DFHPGAI	CSECT	Program autoinstall function	-	03
DFHPGAIT	CSECT	PGAI trace interpretation data	-	03
DFHPGALX	Sample	Program autoinstall exit - PL/I	-	19
DFHPGAOX	Sample	Program autoinstall exit - COBOL	-	19
DFHPGAQ	CSECT	PG domain - inquire/set autoinstall	-	03
DFHPGAQA	DSECT	PGAQ parameter list	OS	-
DFHPGAQM	Macro	PGAQ request	OS	-
DFHPGAQT	CSECT	PGAQ trace interpretation data	-	03
DFHPGAQX	Macro	PGAQ request	11	-
DFHPGAQY	DSECT	PGAQ parameter list	11	-
DFHPGDCD	Source	PG domain anchor block	OS	-
DFHPGDD	CSECT (OCO)	PG domain - define/delete program	-	03
DFHPGDDA	DSECT	PGDD parameter list	OS	-
DFHPGDDM	Macro	PGDD request	OS	-
DFHPGDDT	CSECT (OCO)	PGDD trace interpretation data	-	03
DFHPGDM	CSECT	PG domain - initialize, quiesce, and terminate domain functions	-	03
DFHPGDUF	CSECT (OCO)	PG domain - SDUMP formatter	-	03
DFHPGEX	CSECT (OCO)	PG domain - initialize and terminate exits functions	-	03
DFHPGEXA	DSECT	PGEX parameter list	OS	-
DFHPGEXI	Macro	PGEX inline version of DFHPGEXM	OS	-
DFHPGEXM	Macro	PGEX request	OS	-
DFHPGEXT	Macro (OCO)	PGEX trace interpretation data	-	03
DFHPGGDS	Macro	PG domain - statistics	11	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHPGGDS	Macro	PG domain - statistics	C2	07
DFHPGHM	CSECT (OCO)	PG domain - handle manager services	-	03
DFHPGHMA	DSECT	PGHM parameter list	OS	-
DFHPGHMI	Macro	PGHM inline version of DFHPGHMM	OS	-
DFHPGHMM	Macro	PGHM request	OS	-
DFHPGHMT	CSECT (OCO)	PGHM trace interpretation data	-	03
DFHPGIS	CSECT (OCO)	PG domain - PGIS functions	-	03
DFHPGISA	DSECT	PGIS parameter list	-	11
DFHPGISI	Macro	PGIS inline version of DFHPGHMM	OS	-
DFHPGISM	Macro	PGIS request	-	11
DFHPGIST	CSECT (OCO)	PGIS trace interpretation data	-	03
DFHPGISX	Macro	PGIS request	11	-
DFHPGISY	CSECT	PGIS parameter list	11	-
DFHPGLD	CSECT (OCO)	PG domain - load and release functions	-	03
DFHPGLDA	DSECT	PGLD parameter list	OS	-
DFHPGLDM	Macro	PGLD request	OS	-
DFHPGLDT	CSECT (OCO)	PGLD trace interpretation data	-	03
DFHPGLE	CSECT (OCO)	PG domain - link exec function	-	03
DFHPGLEA	DSECT	PGLE parameter list	OS	-
DFHPGLEM	Macro	PGLE request	OS	-
DFHPGLET	CSECT (OCO)	PGLE trace interpretation data	-	03
DFHPGLK	CSECT (OCO)	PG domain - link and link PLT functions	-	03
DFHPGLKA	DSECT	PGLK parameter list	OS	-
DFHPGLKM	Macro	PGLK request	OS	-
DFHPGLKT	CSECT (OCO)	PGLK trace interpretation data	-	03
DFHPGLU	CSECT (OCO)	PG domain - link URM function	-	03
DFHPGLUA	DSECT	PGLU parameter list	OS	-
DFHPGLUM	Macro	PGLU request	OS	-
DFHPGLUT	CSECT (OCO)	PGLU trace interpretation data	-	03
DFHPGP	Macro	Validate group name for PCT/PPT migrate	11	-
DFHPGPG	CSECT (OCO)	PG domain - initial link function	-	03
DFHPGPGA	DSECT	PGPG parameter list	OS	-
DFHPGPGM	Macro	PGPG request	OS	-
DFHPGPGT	CSECT (OCO)	PGPG trace interpretation data	-	03
DFHPGRE	CSECT (OCO)	PG domain - prepare return function	-	03
DFHPGREA	DSECT	PGRE parameter list	OS	-
DFHPGREM	Macro	PGRE request	OS	-
DFHPGRET	CSECT (OCO)	PGRE trace interpretation data	-	03
DFHPGRP	CSECT (OCO)	PG domain - recovery program	-	03
DFHPGRPT	CSECT (OCO)	PGRP trace interpretation data	-	03
DFHPGST	CSECT (OCO)	PG domain - statistics	-	03
DFHPGTRI	CSECT (OCO)	PG domain - trace interpreter	-	03
DFHPGUE	CSECT (OCO)	PG domain - service requests user exit	-	03
DFHPGXE	CSECT (OCO)	PG domain - prepare XCTL function	-	03
DFHPGXEA	DSECT	PGXE parameter list	OS	-
DFHPGXEM	Macro	PGXE request	OS	-
DFHPGXET	CSECT (OCO)	PGXE trace interpretation data	-	03
DFHPGXM	CSECT (OCO)	PG domain - initialize and terminate transactions functions	-	03
DFHPGXMT	CSECT (OCO)	PGXM trace interpretation data	-	03
DFHPH	Macro	Partition handling macro	11	-
DFHPHN	CSECT	Phonetic code conversion	OS	03
DFHPHP	CSECT	Partition handling program	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHPLARG	DSECT	Generalized domain call parameter list (header, standard fields, responses)	OS	-
DFHPLT	Macro	Program list table	11	-
DFHPLTDS	DSECT	Program list table definition	OS	-
DFHPPFDS	DSECT	KC domain - profile data	OS	-
DFHPRCM	CSECT (OCO)	Partner resource manager command interface	-	03
DFHPRCMA	DSECT	PRCM parameter list	OS	-
DFHPRCMM	Macro	PRCM request	OS	-
DFHPRCMT	CSECT (OCO)	PRCM trace interpretation data	-	03
DFHPRDUF	CSECT (OCO)	Partner resource manager SDUMP formatter SAA communications interface	-	03
DFHPRFS	CSECT (OCO)	Partner resource manager interface to	-	03
DFHPRFSA	DSECT	PRFS parameter list	OS	-
DFHPRFSM	Macro	PRFS request	OS	-
DFHPRFST	CSECT (OCO)	PRFS trace interpretation data	-	03
DFHPRINA	DSECT	PRIN parameter list	OS	-
DFHPRINM	Macro	PRIN request	OS	-
DFHPRINT	Macro	DSECT print control	11	-
DFHPRINU	CSECT (OCO)	PRIN trace interpretation data	-	03
DFHPRIN1	CSECT (OCO)	Partner resource manager initialization management program	-	03
DFHPRIN2	CSECT (OCO)	Partner resource manager initialization subtask program	-	03
DFHPRK	CSECT	3270 print key program	OS	03
DFHPRMCK	Macro	Parameter checking macro	11	-
DFHPROLG	Source	Prologue to DFHENTER	OS	-
DFHPROLM	Source	Acquire LIFO storage application prolog	OS	-
DFHPROLO	Macro	Acquire automatic storage appl prolog	OS	-
DFHPRPT	CSECT (OCO)	Partner resource table (PRT) manager	-	03
DFHPRPTA	DSECT	PRPT parameter list	OS	-
DFHPRPTM	Macro	PRPT request	OS	-
DFHPRPTT	CSECT (OCO)	PRPT trace interpretation data	-	03
DFHPRRP	CSECT (OCO)	Partner resource manager recovery program	-	03
DFHPRRPA	DSECT	PRRP parameter list	OS	-
DFHPRRPM	Macro	PRRP request	OS	-
DFHPRRPT	CSECT (OCO)	PRRP trace interpretation data	-	03
DFHPRSDS	DSECT	Partner static storage area	OS	-
DFHPS	Macro	System spooling interface	OS	-
DFHPSD	Macro	Generate BMS partition set definition	11	-
DFHPSDDS	DSECT	Partition set control block	OS	-
DFHPSGDS	DSECT	Spooler global control block	11	-
DFHPSIP	CSECT	Spooler initialization program	OS	03
DFHPSP	CSECT	System spooling interface program	OS	03
DFHPSPCK	CSECT	System spooling subsystem activator	OS	03
DFHPSPDW	CSECT	System spooling interface, DWE processor	OS	03
DFHPSPSS	CSECT	System spooling JES interface subtask	OS	03
DFHPSPST	CSECT	System spooling JES interface control	OS	03
DFHPSSVC	CSECT	System spooling interface, retrieve a data set name	OS	03
DFHPTDUF	CSECT (OCO)	Program control table SDUMP formatter	-	03
DFHPUPAB	CSECT	CSDUP - initialize RDO parameter fields and address list (DFHPUPA)	OS	03
DFHPUPAC	CSECT	CSDUP - initialize RDO parameter fields and address list (DFHPUPA)	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHPUPB	CSECT	CSDUP - RDO parameter utility program, batch environment (DFHPUP batch)	-	03
DFHPUPC	CSECT	RDO parameter utility program, CICS environment (DFHPUP CICS)	-	03
DFHPUPDB	CSECT	CSDUP - default parameter values lookup (DFHPUPD batch)	OS	03
DFHPUPDC	CSECT	RDO parameter utility - default parameter values lookup (DFHPUPD CICS)	OS	03
DFHPUPXB	CSECT	CSDUP - language table referencing functions (DFHPUPX batch)	OS	03
DFHPUPXC	CSECT	RDO parameter utility - language table referencing functions (DFHPUPX CICS)	OS	03
DFHP3270	CSECT	3270 print function support	OS	03
DFHQRY	CSECT	Query transaction	OS	03
DFHQSSS	CSECT (OCO)	Qualified subsystem services	-	03
DFHRCEX	CSECT	Recovery control enable exit	OS	03
DFHRCNO	Other	Used by DFHSTART cataloged procedure	19	-
DFHRCSDS	DSECT	Recovery control static storage	OS	-
DFHRCYES	Other	Used by DFHSTART cataloged procedure	19	-
DFHRDDUF	CSECT	Resource definition recovery offline dump exit	-	03
DFHRDJPN	CSECT (OCO)	CSD utilities - RDL for Japanese language feature upgrade	-	03
DFHREGS	Macro	Standard register name definition	11	-
DFHREQ	Macro	Attention ID coding macro	11	-
DFHREST	CSECT	User-replaceable restart program	19	03
DFHRITRI	CSECT	RMI trace interpretation routine	-	03
DFHRKB	CSECT	3270 release keyboard program	OS	03
DFHRLR	CSECT	BMS route list resolution	OS	-
DFHRLRA$	CSECT	BMS route list resolution (standard)	OS	03
DFHRLR1$	CSECT	BMS route list resolution (full)	OS	03
DFHRMCAL	Macro	Resource manager call	11	-
DFHRMCD	CSECT	Recovery manager client directory	-	03
DFHRMCDA	CSECT	RMCD parameter list	OS	-
DFHRMCDM	Macro	RMCD request	OS	-
DFHRMCDT	DSECT	RMCD translate tables	-	03
DFHRMCD1	CSECT	RM client directory class initialization	-	03
DFHRMCD2	CSECT	RM client directory class quiesce proc	-	03
DFHRMCI2	CSECT	RM client directory set gate procedure	-	03
DFHRMCI3	CSECT	RM client directory wait for client proc	-	03
DFHRMCI4	CSECT	RM client directory send procedure	-	03
DFHRMDEA	CSECT	RMDE parameter list	OS	-
DFHRMDEM	Macro	RMDE request	OS	-
DFHRMDET	DSECT	RMDE translate tables	-	03
DFHRMDM	CSECT	Recovery manager domain management	-	03
DFHRMDMA	CSECT	RMDM parameter list	OS	-
DFHRMDMM	Macro	RMDM request	OS	-
DFHRMDMT	DSECT	RMDM translate tables	-	03
DFHRMDU0	CSECT	RMCI dump formatting	-	03
DFHRMDU2	CSECT	RMDU start work token browse procedure	-	03
DFHRMDU3	CSECT	RMDU get next work token procedure	-	03
DFHRMDU4	CSECT	RMDU end work token browse procedure	-	03
DFHRMDU5	CSECT		-	03
DFHRMGDS	CSECT	Recovery manager global statistics	11	-
DFHRMGDS	CSECT	Recovery manager global statistics	C2	07

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHRMKDA	CSECT	RMKD parameter list	OS	-
DFHRMKDM	Macro	RMKD request	OS	-
DFHRMKDT	DSECT	RMKD translate tables	-	03
DFHRMKPA	CSECT	RMKP parameter list	OS	-
DFHRMKPM	Macro	RMKP request	OS	-
DFHRMKPT	DSECT	RMKP translate tables	-	03
DFHRMLKQ	CSECT	RMLK quiesce procedure	-	03
DFHRMLKT	DSECT	RMLK translate tables	-	03
DFHRMLK1	CSECT	RMLK initialize class procedure	-	03
DFHRMLK2	CSECT	RMLK initiate recovery2 procedure	-	03
DFHRMLK3	CSECT	RMLK inquire logname procedure	-	03
DFHRMLK4	CSECT	RMLK clear pending2 procedure	-	03
DFHRMLK5	CSECT	RMLK collect statistics procedure	-	03
DFHRMLN	CSECT	RMLN gate handler module	-	03
DFHRMLNA	CSECT	RMLN parameter list	OS	-
DFHRMLNM	Macro	RMLN request	OS	-
DFHRMLNT	DSECT	RMLN translate table	-	03
DFHRMLSD	CSECT	Recovery Manager LinkSet class declaration	-	03
DFHRMLSF	CSECT	RMLS inquire awaiting forget procedure	-	03
DFHRMLSO	CSECT	RMLS commit procedure	-	03
DFHRMLSP	CSECT	RMLS prepare procedure	-	03
DFHRMLSS	CSECT	RMLS shunt procedure	-	03
DFHRMLSU	CSECT	RMLS unshunt procedure	-	03
DFHRML1D	CSECT	RMLK deliver data procedure	-	03
DFHRMNM	CSECT	Recovery Manager Lognames class	-	03
DFHRMNMA	CSECT	RMNM parameter list	OS	-
DFHRMNMM	Macro	RMNM request	OS	-
DFHRMNMT	DSECT	RMNM translate tables	-	03
DFHRMNM1	CSECT	RMNM initialize class procedure	-	03
DFHRMNS1	CSECT	RMNS initialize class procedure	-	03
DFHRMNS2	CSECT	RMNS quiesce procedure	-	03
DFHRMOFI	CSECT	RMOF initialize procedure	-	03
DFHRMOT	CSECT	RMOT CDURUN and Gate Module	-	03
DFHRMOTT	CSECT		-	03
DFHRMREA	CSECT	RMRE parameter list	OS	-
DFHRMREM	Macro	RMRE request	OS	-
DFHRMRET	DSECT	RMRE translate tables	-	03
DFHRMRO	CSECT	RM resource owner class	-	03
DFHRMROA	CSECT	RMRO parameter list	OS	-
DFHRMROM	Macro	RMRO request	OS	-
DFHRMROO	CSECT	RMRO forgotten procedure	-	03
DFHRMROS	CSECT	RMRO shunt procedure	-	03
DFHRMROT	CSECT	RMRO translate tables	-	03
DFHRMROU	CSECT	RMRO unshunt procedure	-	03
DFHRMROV	CSECT	RMRO avail procedure	-	03
DFHRMRO1	CSECT	RMRO initialize class procedure	-	03
DFHRMRO2	CSECT	RMRO start back out procedure	-	03
DFHRMRO3	CSECT	RMRO deliver back out data procedure	-	03
DFHRMRO4	CSECT	RMRO end back out procedure	-	03
DFHRMRS	CSECT	RM RMC CDURUN and Gate Module	-	03
DFHRMR1D	CSECT	RMRO deliver data procedure	-	03
DFHRMR1E	CSECT	RMRO end delivery procedure	-	03
DFHRMR1K	CSECT	RMRO take keypoint procedure	-	03
DFHRMR1S	CSECT	RMRO start delivery procedure	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHRMSL	CSECT	RM system log class	-	03
DFHRMSLA	CSECT	RMSL parameter list	OS	-
DFHRMSLF	CSECT	RMSL force procedure	-	03
DFHRMSLJ	CSECT	RMSL notify disjoint chains procedure	-	03
DFHRMSLL	CSECT	RMSL close chain procedure	-	03
DFHRMSLM	Macro	RMSL request	OS	-
DFHRMSLO	CSECT	RMSL open chain procedure	-	03
DFHRMSLT	CSECT	RMSL translate tables	-	03
DFHRMSLV	CSECT	RMSL move chain procedure	-	03
DFHRMSLW	CSECT	RMSL write procedure	-	03
DFHRMSL1	CSECT	RMSL initialize class procedure	-	03
DFHRMSL2	CSECT	RMSL start chain browse procedure	-	03
DFHRMSL3	CSECT	RMSL chain browse read procedure	-	03
DFHRMSL4	CSECT	RMSL end chain browse procedure	-	03
DFHRMSL5	CSECT	RMSL restart procedure	-	03
DFHRMSL6	CSECT	RMSL schedule keypoint procedure	-	03
DFHRMSL7	CSECT	RMSL take keypoint procedure	-	03
DFHRMST	CSECT	RM statistics class	-	03
DFHRMST1	CSECT	RMST initialize class procedure	-	03
DFHRMSY	CSECT	Resource Manager resynchronization program	-	03
DFHRMTRI	CSECT	Offline trace formatting interpretation routine parameter list	-	03
DFHRMUC	CSECT	Resource Manager create UOW	-	03
DFHRMUO	CSECT	Resource Manager commit UOW	-	03
DFHRMUW	CSECT	Resource Manager unit of work class	-	03
DFHRMUTL	CSECT	Resource Manager batch utility program	-	03
DFHRMUWA	CSECT	RMUW parameter list	OS	-
DFHRMUWB	CSECT	RMUW deliver backout procedure	-	03
DFHRMUWE	CSECT	RMUW unshunt reply procedure	-	03
DFHRMUWF	CSECT	RMUW force procedure	-	03
DFHRMUWH	CSECT	RMUW hold procedure	-	03
DFHRMUWI	Macro	RMUWI inquire UOQ ID	OS	-
DFHRMUWJ	CSECT	RMUW force heurism procedure	-	03
DFHRMUWL	CSECT	RMUW forget links procedure	-	03
DFHRMUWM	Macro	RMUW request	OS	-
DFHRMUWN	CSECT	RMUW unshunt procedure	-	03
DFHRMUWP	CSECT	RMUW process avail procedure	-	03
DFHRMUWQ	CSECT	RMUW process indoubt resolution procedure	-	03
DFHRMUWS	CSECT	RMUW record decision procedure	-	03
DFHRMUWT	DSECT	RM unit of work class (timeout)	-	03
DFHRMUWU	CSECT	RMUW set local lu name procedure	-	03
DFHRMUWV	CSECT	RMUW avail procedure	-	03
DFHRMUWW	CSECT	RMUW write procedure	-	03
DFHRMUW0	CSECT	RMUW release procedure	-	03
DFHRMUW1	CSECT	RMUW initialize class procedure	-	03
DFHRMUW2	CSECT	RMUW collect statistics procedure	-	03
DFHRMUW3	CSECT	RMUW inquire work token procedure	-	03
DFHRMUXD	DSECT	Define parts of UOW objects accessible by inline macros	OS	-
DFHRMU1C	CSECT	RMUW set chain token procedure	-	03
DFHRMU1D	CSECT	RMUW deliver data procedure	-	03
DFHRMU1E	CSECT	RMUW end delivery procedure	-	03
DFHRMU1F	CSECT	RMUW wait timeout notify procedure	-	03
DFHRMU1G	CSECT	RMUW	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHRMU1J	CSECT	RMUW inquire disjoint chains procedure	-	03
DFHRMU1K	CSECT	RMUW take keypoint procedure	-	03
DFHRMU1L	CSECT	xphP force purge inhibit query gate	-	03
DFHRMU1N	CSECT	RMU1 force purge query procedure	-	03
DFHRMU1Q	CSECT	TISR notify gate	-	03
DFHRMU1R	CSECT	RMUW restart procedure	-	03
DFHRMU1S	CSECT	RMUW start delivery procedure	-	03
DFHRMU1U	CSECT	RMUW process restart procedure	-	03
DFHRMU1V	CSECT	RMUW request wait timeout procedure	-	03
DFHRMU1W	CSECT	RMUW cancel wait timeout procedure	-	03
DFHRMVP1	CSECT	RMVP initialize class procedure	-	03
DFHRMWTA	CSECT	RMWT parameter list	OS	-
DFHRMWTI	Macro	Supports the Inquire_work_token and Set_work_token of RMWT CDURUN interface	OS	-
DFHRMWTM	Macro	RMWT request	OS	-
DFHRMWTT	DSECT	RMWT translate tables	-	03
DFHRMXNE	CSECT	RMXN reattach procedure	-	03
DFHRMXN2	CSECT	RMXN schedule keypoint procedure	-	03
DFHRMXN3	CSECT	RMXN keypoint transaction	-	03
DFHRMXN4	CSECT	RMXN restart procedure	-	03
DFHRMXN5	CSECT	RMXN inc trandef statistic procedure	-	03
DFHROINA	CSECT	ROIN parameter list	OS	-
DFHROINM	Macro	ROIN request	OS	-
DFHROINT	DSECT	ROIN translate tables	OS	03
DFHRPAL	CSECT (OCO)	ONC RPC Feature alias list	-	03
DFHRPALT	DSECT	RPAL translate tables	-	03
DFHRPAS	CSECT (OCO)	ONC RPC alias main program	-	03
DFHRPCC	CSECT (OCO)	RPCC parameter list	-	03
DFHRPCB	Macro	Extension to DL/I PCB control block - contains ISC information about PCB	OS	-
DFHRPCDH	CSECT	RPPC caller DFHRPCC parameter list	-	08
DFHRPCDO	CSECT	RPPC caller DFHRPCC parameter list	-	07
DFHRPC0A	CSECT (OCO)	CRPC dataset list processing	-	03
DFHRPC0B	CSECT (OCO)	CRPC common subroutines	-	03
DFHRPC0D	CSECT (OCO)	CRPC register remote procedures	-	03
DFHRPC0E	CSECT (OCO)	CRPC register remote procedures	-	03
DFHRPC01	CSECT (OCO)	CRPC initial processing	-	03
DFHRPC03	CSECT (OCO)	CRPC manage feature dataset	-	03
DFHRPC04	CSECT (OCO)	CRPC disable processing	-	03
DFHRPC05	CSECT (OCO)	CRPC manage feature dataset	-	03
DFHRPC06	CSECT (OCO)	CRPC update feature	-	03
DFHRPC08	CSECT (OCO)	CRPC ONC RPC feature	-	03
DFHRPC09	CSECT (OCO)	ONC RPC registration table management	-	03
DFHRPC10	CSECT (OCO)	CRPC alias list processing	-	03
DFHRPC4C	CSECT (OCO)	ONC RPC initialization	-	03
DFHRPC42	CSECT (OCO)	CRPC enable request processing	-	03
DFHRPDUF	CSECT (OCO)	System dump formatting routine for ONC/RPC	OS	03
DFHRPMS	CSECT (OCO)	ONC RPC feature server controller	-	03
DFHRPRDH	CSECT	RPRSC parameter list	-	08
DFHRPRDO	CSECT	RPRSC parameter list	-	07
DFHRPRP	CSECT (OCO)	ONC RPC feature RPC caller	-	03
DFHRPRPT	CSECT (OCO)	RPRP call structured parameter list	-	03
DFHRPTRI	CSECT (OCO)	ONC RPC feature trace interpretation	-	03
DFHRPTRU	CSECT (OCO)	ONC RPC task-related user exit	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHRPUCH	CSECT	Constants used by user replaceable programs	-	08
DFHRPUCO	CSECT	Constants used by user replaceable programs	-	07
DFHRP0	CSECT (OCO)	BMS mapset for CRPC main panels	-	03
DFHRP0H	CSECT (OCO)	CRPC DFHRP0 help panels	-	03
DFHRST	Macro	DBCTL XRF recoverable service table	11	-
DFHRTC	CSECT	CRTE cancel command processor	OS	03
DFHRTE	CSECT	Transaction routing program	OS	03
DFHRTSU	CSECT	Surrogate terminal interface program	-	03
DFHRTSUA	CSECT	RTSU parameter list	OS	-
DFHRTSUI	CSECT	Provide Assign/Relay relay link functions of DFHRTSU	OS	-
DFHRTSUM	Macro	RTSU request	OS	-
DFHRTSUT	DSECT	RTSU translate tables	-	03
DFHRTTRI	CSECT	ISC transaction routing (APRT) trace interpreter	OS	03
DFHRTTR1	CSECT	RTSU trace interpretation	-	03
DFHRXAST	CSECT		-	03
DFHRXDM	CSECT	RX Domain Management	-	03
DFHRXDMA	CSECT	RXDM interface parameter area	OS	-
DFHRXDMM	Macro	DFHRXDM interface macro	OS	-
DFHRXDMT	CSECT		-	03
DFHRXDUF	CSECT	RX Domain Dump Formatting	-	03
DFHRXSVC	CSECT	RX Domain Management	-	03
DFHRXTRI	CSECT	DFHRXTRI Design	-	03
DFHRXUW	CSECT	RX Domain UOW Manager	-	03
DFHRXUWA	Other	RXUW interface parameter area	OS	-
DFHRXUWM	Macro	RXUW interface macro	OS	-
DFHRXUWT	CSECT		-	03
DFHRXXMA	Other	RXXM interface parameter area	OS	-
DFHRXXMM	Macro	DFHRXXM interface macro	OS	-
DFHRXXMT	CSECT		-	03
DFHRXXRG	CSECT		-	03
DFHRXXRM	CSECT		-	03
DFHRZDM	CSECT		-	03
DFHRZDUF	CSECT	RequestStreams remote join interface	-	03
DFHRZIX	CSECT		-	03
DFHRZJN	CSECT		-	03
DFHRZLN	CSECT		-	03
DFHRZNR2	CSECT		-	03
DFHRZOFI	CSECT		-	03
DFHRZRG2	CSECT		-	03
DFHRZRJ	CSECT	RequestStreams remote join interface	-	03
DFHRZRJT	CSECT		-	03
DFHRZRM	CSECT	RZRM Gate Module for RM RO callback	-	03
DFHRZRS1	CSECT		-	03
DFHRZRT	CSECT	RZRT CDURUN and Gate Module	-	03
DFHRZRT1	CSECT		-	03
DFHRZRT2	CSECT		-	03
DFHRZSO	CSECT		-	03
DFHRZSOT	CSECT		-	03
DFHRZSO1	CSECT		-	03
DFHRZTA	CSECT		-	03
DFHRZTAT	CSECT		-	03
DFHRZTCX	CSECT		-	03
DFHRZTRI	CSECT	RequestStreams Trace interpretation	-	03
DFHRZTR1	CSECT		-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHRZVP1	CSECT		-	03
DFHRZXM	CSECT	RequestStreams XM Attach Client	-	03
DFHR2TRI	CSECT		-	03
DFHSAADS	DSECT	Storage accounting area	11	-
DFHSABDS	DSECT	Subsystem anchor block	OS	-
DFHSAIQ	CSECT (OCO)	AP domain - system data inquire and set	-	03
DFHSAIQT	CSECT (OCO)	SAIQ trace interpretation data	-	03
DFHSAIQX	Macro	SAIQ request	11	-
DFHSAIQY	DSECT	SAIQ parameter list	11	-
DFHSAXDF	CSECT	DU domain - transaction dump formatter for system areas (CSA, TCA, and so on)	OS	03
DFHSC	Macro	Storage service request	11	-
DFHSCAA	CSECT	Language Environment - set common anchor area	OS	03
DFHSCALL	Macro	EXEC interface call macro for CICSPlex SM commands in assembler-language pgms	11	-
DFHSCCOS	Symbolic	Storage control class of storage	OS	-
DFHSDGDS	DSECT	System dump global statistics	11	-
DFHSDGDS	DSECT	System dump global statistics	C2	07
DFHSDMP	Macro	SDUMP parameter area and MD=L expansion	OS	-
DFHSDRDS	DSECT	System dump statistics by dump code	11	-
DFHSDRDS	DSECT	System dump statistics by dump code	C2	07
DFHSFP	CSECT	Sign-off program	OS	03
DFHSFTC	CSECT		OS	-
DFHSGTIM	CSECT		OS	-
DFHSHDM	CSECT	SH Domain Management	-	03
DFHSHDUF	CSECT	SH Domain Dump Formatting	-	03
DFHSHOFI	CSECT		-	03
DFHSHPR	CSECT	SHPR CDURUN and Gate Module	-	03
DFHSHPRT	CSECT		-	03
DFHSHRE1	CSECT		-	03
DFHSHRM	CSECT	SHRM CDURUN and Gate Module	-	03
DFHSHRQ	CSECT	Scheduler Services - Request Queue	-	03
DFHSHRQA	Other	SHRQ interface parameter area	OS	-
DFHSHRQM	Macro	DFHSHRQ interface macro	OS	-
DFHSHRQT	CSECT		-	03
DFHSHRQ1	CSECT		-	03
DFHSHRR	CSECT	SHRR CDURUN and Gate Module	-	03
DFHSHRRP	CSECT		-	03
DFHSHRRT	CSECT		-	03
DFHSHRSP	CSECT		-	03
DFHSHRT	CSECT	SHRT CDURUN and Gate Module	-	03
DFHSHRTT	CSECT		-	03
DFHSHRT1	CSECT		-	03
DFHSHRT2	CSECT		-	03
DFHSHSY	CSECT	Component modules	-	03
DFHSHTC	CSECT		OS	-
DFHSHTI	CSECT		-	03
DFHSHTRI	CSECT	SH Domain Trace Interpretation	-	03
DFHSHVP1	CSECT		-	03
DFHSHWPL	DSECT	File control SHOWCAT parameter list	OS	-
DFHSHXM	CSECT	Scheduler Services XM Attach Client	-	03
DFHSIA1	CSECT	System initialization - module A1	OS	03
DFHSIB1	CSECT	System initialization - module B1	OS	03
DFHSIB1A	Source	DFHSIB1 pre-nucleus load routines	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHSIB1B	Source	DFHSIB1 nucleus load routine	OS	-
DFHSIB1C	Source	DFHSIB1 post-nucleus load routine	OS	-
DFHSIB1D	Source	DFHSIB1 subroutines	OS	-
DFHSICOM	Macro	System initialization definitions	OS	-
DFHSIC1	CSECT	System initialization - module C1	OS	03
DFHSID1	CSECT	System initialization - module D1	OS	03
DFHSIF1	CSECT	System initialization - module F1	OS	03
DFHSIG1	CSECT	System initialization - module G1	OS	03
DFHSIH1	CSECT	System initialization - module H1	OS	03
DFHSII1	CSECT	System initialization - module I1	OS	03
DFHSIJ1	CSECT	System initialization - module J1	OS	03
DFHSIPD	Macro	Generate system initialization communication area	OS	-
DFHSIPDS	DSECT	SIP communication area	OS	-
DFHSIPLT	CSECT	System initialization - PLT processor	OS	03
DFHSIT	Macro	System initialization table	11	-
DFHSIT$$	Sample	Default system initialization table	19	03
DFHSIT6$	Sample	System initialization table	19	03
DFHSJAS	CSECT	SJ Assembler routines for DFHSJCS	-	03
DFHSJCS@	CSECT	Autocall SCEEOBJ	-	03
DFHSJDM	CSECT	SJ SJVM Domain	-	03
DFHSJDUF	CSECT	SJ SJVM Domain	-	03
DFHSJGDS	DSECT	Jvmpool Global Statistics	11	07
DFHSJIIN	CSECT	SJ JVM Domain	-	03
DFHSJINT	CSECT		-	03
DFHSJIS	CSECT	SJ JVM Domain	-	03
DFHSJIST	CSECT		-	03
DFHSJJ8H	CSECT		-	08
DFHSJJ8O	CSECT	SJ JVM Domain	OS	03
DFHSJST	CSECT	(SOCKETS) Statistics functions	-	03
DFHSJTRI	CSECT	SJ SJVM Domain	-	03
DFHSK	Macro	Subtasking interface	OS	-
DFHSKC	CSECT	Subtask control program	OS	03
DFHSKE	CSECT	Subtask execution program	OS	03
DFHSKM	CSECT	Subtask manager	OS	03
DFHSKR	Macro	Generate SKR table entries in SIT	11	-
DFHSKTSK	CSECT	General purpose subtask entry point	OS	03
DFHSLDC	DSECT	System logical device code table	11	-
DFHSMAD	CSECT (OCO)	SM domain - add/delete subpool	-	03
DFHSMADA	DSECT	SMAD parameter list	OS	-
DFHSMADM	Macro	SMAD request	OS	-
DFHSMADT	CSECT (OCO)	SMAD trace interpretation data	-	03
DFHSMAFA	DSECT	SMAF parameter list	OS	-
DFHSMAFT	CSECT (OCO)	SMAF trace interpretation data	-	03
DFHSMAR	CSECT (OCO)	SM domain - handle functions	-	03
DFHSMART	CSECT (OCO)	SMAR trace interpretation data	-	03
DFHSMCK	CSECT (OCO)	SM domain - storage checking/recovery	-	03
DFHSMCKA	DSECT	SMCK parameter list	OS	-
DFHSMCKM	Macro	SMCK request	OS	-
DFHSMCKT	CSECT (OCO)	SMCK trace interpretation data	-	03
DFHSMDDS	DSECT	SM domain - storage statistics for domain subpools	11	-
DFHSMDDS	DSECT	SM domain - storage statistics for domain subpools	C2	07
DFHSMDM	CSECT (OCO)	SM domain - initialization/termination	-	03
DFHSMDUF	CSECT (OCO)	SDUMP formatter for SM domain	-	03
DFHSMFDS	DSECT	SMF header and product section (JC/MN/ST)	11	07

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHSMGF	CSECT (OCO)	SM domain - getmain/freemain	-	03
DFHSMGFA	DSECT	SMGF parameter list	OS	-
DFHSMGFI	Macro	SM domain - inline getmain/freemain	OS	-
DFHSMGFM	Macro	SMGF request	OS	-
DFHSMGFT	CSECT (OCO)	SMGF trace interpretation data	-	03
DFHSMMCA	DSECT	SMMC parameter list	OS	-
DFHSMMCI	CSECT (OCO)	SM domain - macro-compatibility initialize	-	03
DFHSMMCM	Macro	SMMC request	OS	-
DFHSMMCT	CSECT (OCO)	SMMC trace interpretation data	-	03
DFHSMMCX	Macro	SMMC request (XPI)	11	-
DFHSMMCY	DSECT	SMMC parameter list (XPI)	11	-
DFHSMMC2	CSECT (OCO)	SM domain - macro-compatibility system freemain functions	-	03
DFHSMMF	CSECT (OCO)	SM domain - macro-compatibility freemain interface	-	03
DFHSMMG	CSECT (OCO)	SM domain - macro-compatibility getmain interface	-	03
DFHSMNTA	DSECT	SMNT parameter list	OS	-
DFHSMNTM	Macro	SMNT request	OS	-
DFHSMNTT	CSECT (OCO)	SMNT trace interpretation data	-	03
DFHSMPE	Other	Cataloged procedure to execute SMP/E	02	-
DFHSMPP	CSECT (OCO)	SM domain - pagepool manager functions 1	-	03
DFHSMPPT	CSECT (OCO)	SMPP trace interpretation data	-	03
DFHSMPQ	CSECT (OCO)	SM domain - pagepool manager functions 2	-	03
DFHSMPQT	CSECT (OCO)	SMPQ trace interpretation data	-	03
DFHSMPT	Macro	SMP/E control card generator	11	-
DFHSMSCP	CSECT (OCO)	Storage control program	-	03
DFHSMSDS	DSECT	SM domain - storage statistics for DSAs	11	-
DFHSMSDS	DSECT	SM domain - storage statistics for DSAs	C2	07
DFHSMSQ	CSECT (OCO)	SM domain - suspend queue manager function	-	03
DFHSMSQT	CSECT (OCO)	SMSQ trace interpretation data	-	03
DFHSMSR	CSECT (OCO)	SM domain - services	-	03
DFHSMSRA	DSECT	SMSR parameter list	OS	-
DFHSMSRI	CSECT	SM domain - in-line INQUIRE_ACCESS	OS	-
DFHSMSRM	Macro	SMSR request	OS	-
DFHSMSRT	CSECT (OCO)	SMSR trace interpretation data	-	03
DFHSMSRX	Macro (OCO)	SMSR request (XPI)	11	-
DFHSMSRY	DSECT (OCO)	SMSR parameter list	11	-
DFHSMST	CSECT (OCO)	SM domain - statistics collection	-	03
DFHSMSU	CSECT (OCO)	Subspace manager	-	03
DFHSMSUT	CSECT (OCO)	Subspace manager trace interpretation data	-	03
DFHSMSVC	CSECT (OCO)	SM domain - authorized service routine	-	03
DFHSMSY	CSECT (OCO)	SM domain - system task	-	03
DFHSMTAB	CSECT	CICSPLex SM commands language table	-	03
DFHSMTDS	DSECT	SM domain - storage statistics for task subpools	11	-
DFHSMTDS	DSECT	SM domain - storage statistics for task subpools	C2	-
DFHSMTDS	DSECT	SM domain - storage statistics for task subpools	P2	-
DFHSMTRI	CSECT (OCO)	Trace interpreter for SM domain	-	03
DFHSMUTL	CSECT	SM Catalog Update Program	OS	03
DFHSMXDF	CSECT (OCO)	Transaction dump - task subpools	-	03
DFHSNAS	CSECT	create signon/sign-off ATI sessions	-	03
DFHSNEP	Macro	Node error program generator	11	-
DFHSNEPH	Macro	NEP inner macro	11	-
DFHSNET	Macro	Node error table generator	11	-
DFHSNEX	Macro	Signon extension block generator	11	-
DFHSNEXD	DSECT	Signon extension to TCTTE	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHSNGND	DSECT	CEGN parameter list	OS	-
DFHSNGSD	DSECT	GNTRAN parameter list	11	-
DFHSNGSH	DSECT	GNTRAN parameter list (C/370)	-	08
DFHSNGSL	DSECT	GNTRAN parameter list (PL/I)	-	17
DFHSNGSO	DSECT	GNTRAN parameter list (COBOL)	C2	-
DFHSNLE	CSECT	Signon large screens map set	OS	03
DFHSNLK	CSECT (OCO)	Signon large screens map set	-	03
DFHSNMIG	CSECT	Signon table migration utility	OS	03
DFHSNNFY	CSECT	RACF CICS segment notify exit	OS	03
DFHSNP	CSECT	Signon program	OS	03
DFHSNPTO	CSECT	CICS segment (RACF) TIMEOUT keyword print exit routine	-	03
DFHSNPU	CSECT	Preset userid signon/sign-off	-	03
DFHSNSC	CSECT	Timeout transaction (CESC) scheduler	-	03
DFHSNSCA	CSECT	SNSC parameter list	OS	-
DFHSNSCM	Macro	SNSC requests	OS	-
DFHSNSE	CSECT	Signon small screens map set	OS	03
DFHSNSG	CSECT	Surrogate terminal signon/off	-	03
DFHSNSGI	Macro	Surrogate terminals sign-on and signoff requests	OS	-
DFHSNSK	CSECT (OCO)	Signon small screens map set	-	03
DFHSNSTA	DSECT	ISC/IRC attach-time statistics area	OS	-
DFHSNSU	CSECT	Session userid signon/sign-off	-	03
DFHSNTRI	CSECT	SN trace interpreter	-	03
DFHSNTU	CSECT	Terminal userid signon/sign-off	-	03
DFHSNUS	CSECT (OCO)	US domain - local and remote signon	-	03
DFHSNUSA	DSECT	SNUS parameter list	OS	-
DFHSNUSM	Macro	SNUS macro	OS	-
DFHSNUST	CSECT (OCO)	SNUS trace interpretation data	-	03
DFHSNVCL	CSECT	RACF CICS segment OPCLASS validation exit	OS	03
DFHSNVID	CSECT	RACF CICS segment OPIDENT validation exit	OS	03
DFHSNVPR	CSECT	RACF CICS segment OPPRTY validation exit	OS	03
DFHSNVTO	CSECT	RACF CICS segment TIMEOUT validation exit	OS	03
DFHSNXR	CSECT (OCO)	XRF reflecting signon state	-	03
DFHSNXRA	DSECT	SNXR parameter list	OS	-
DFHSNXRM	Macro	SNXR requests	OS	-
DFHSNXRT	CSECT (OCO)	SNXR trace interpretation data	-	03
DFHSOAD	CSECT	SO Domain - SOAD gate functions	-	03
DFHSOADT	CSECT		-	03
DFHSOCBT	CSECT		-	03
DFHSOCK	CSECT	Sockets send/receive/close	-	03
DFHSOCKT	CSECT		-	03
DFHSODM	CSECT	Sockets Domain Initialization	-	03
DFHSODUF	CSECT	Sockets Domain Dump Formatting	-	03
DFHSOGDS	DSECT	Sockets Global Statistics	11	07
DFHSOGH@	DSECT		-	03
DFHSOIS	CSECT	Sockets Domain Inquire/Set	-	03
DFHSOIST	CSECT		-	03
DFHSOL	CSECT	Sockets Domain Listener Task	-	03
DFHSOLS	CSECT	Sockets Listener	-	03
DFHSOLST	CSECT		-	03
DFHSOLX	CSECT	Sockets Domain Asynchronous exit routine	-	03
DFHSOPI	CSECT	SO Domain CEEPIPI service routines	-	03
DFHSORD	CSECT	SO Domain Sockets Register/Deregister	-	03
DFHSORDS	Other	SO Domain TCPIP Service Statistics	11	07

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHSORDT	CSECT		-	03
DFHSORT	Macro	Auxiliary sort	11	-
DFHSOSE	CSECT	Sockets Domain Secure Sockets Layer	-	03
DFHSOSET	CSECT		-	03
DFHSOSK@	CSECT		-	03
DFHSOSKO	CSECT		-	03
DFHSOST	CSECT	Sockets Statistics Functions	-	03
DFHSOS00	CSECT		-	03
DFHSOS01	CSECT		-	03
DFHSOS02	CSECT		-	03
DFHSOS03	CSECT		-	03
DFHSOS04	CSECT		-	03
DFHSOS05	CSECT		-	03
DFHSOS06	CSECT		-	03
DFHSOS07	CSECT		-	03
DFHSOS08	CSECT		-	03
DFHSOS09	CSECT		-	03
DFHSOS10	CSECT		-	03
DFHSOS11	CSECT		-	03
DFHSOS12	CSECT		-	03
DFHSOS13	CSECT		-	03
DFHSOS14	CSECT		-	03
DFHSOS15	CSECT		-	03
DFHSOS16	CSECT		-	03
DFHSOS17	CSECT		-	03
DFHSOS18	CSECT		-	03
DFHSOS19	CSECT		-	03
DFHSOS20	CSECT		-	03
DFHSOS21	CSECT		-	03
DFHSOS22	CSECT		-	03
DFHSOS23	CSECT		-	03
DFHSOTB	CSECT	SO Domain SOTB Gate Functions	-	03
DFHSOTBT	CSECT		-	03
DFHSOTI	CSECT	Sockets Timer	-	03
DFHSOTRI	CSECT	Sockets Domain Trace Interpretation	-	03
DFHSOUE	CSECT	Sockets Domain User Exit Services	-	03
DFHSOXM	CSECT	Sockets Attach Client	-	03
DFHSP	Macro	Syncpoint service request	11	-
DFHSPBAB	CSECT		-	03
DFHSPBAC	CSECT		-	03
DFHSPBAE	CSECT		-	03
DFHSPDBB	CSECT		OS	03
DFHSPDBC	CSECT		OS	03
DFHSPDBE	CSECT		OS	03
DFHSPDHB	CSECT		-	03
DFHSPDHC	CSECT		-	03
DFHSPDHE	CSECT		-	03
DFHSPEJB	CSECT		-	03
DFHSPEJC	CSECT		-	03
DFHSPEJE	CSECT		-	03
DFHSPFIB	CSECT	CSDUP - cross-keyword validation for files	OS	03
DFHSPFIC	CSECT	RDO - cross-keyword validation for files	OS	03
DFHSPFIE	CSECT	RDO file definition validation	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHSPKCB	CSECT	CSDUP - cross-keyword validation for transactions and profiles	OS	03
DFHSPKCC	CSECT	RDO - cross-keyword validation for transactions and profiles	OS	03
DFHSPKCE	CSECT	RDO txn control definition validation	OS	03
DFHSPLMB	CSECT	RDO JournalModel definition validation	-	03
DFHSPLMC	CSECT	RDO JournalModel definition validation	-	03
DFHSPLME	CSECT	RDO JournalModel definition validation	-	03
DFHSPLSB	CSECT	CSDUP - cross-keyword validation for LSR pools	OS	03
DFHSPLSC	CSECT	RDO - cross-keyword validation for LSR pools	OS	03
DFHSPLSE	CSECT	RDO - Lsrpool definition validation	OS	03
DFHSPNQB	CSECT		OS	03
DFHSPNQC	CSECT		OS	03
DFHSPNQE	CSECT		OS	03
DFHSPOPB	CSECT		-	03
DFHSPOPC	CSECT		-	03
DFHSPOPE	CSECT		-	03
DFHSPPCB	CSECT	CSDUP - cross-keyword validation for programs, map sets, and partition sets	OS	03
DFHSPPCC	CSECT	RDO - cross-keyword validation for programs, map sets, and partition sets	OS	03
DFHSPPCE	CSECT	RDO - program definition validation	OS	03
DFHSPPNB	CSECT	CSDUP - cross-keyword validation for partners	OS	03
DFHSPPNC	CSECT	RDO - cross-keyword validation for partners	OS	03
DFHSPPNE	CSECT	RDO - partner definition validation	OS	03
DFHSPSOB	CSECT		-	03
DFHSPSOC	CSECT		-	03
DFHSPSOE	CSECT		-	03
DFHSPTCB	CSECT	CSDUP - cross-keyword validation for terminals	OS	03
DFHSPTCC	CSECT	RDO - cross-keyword validation for terminals	OS	03
DFHSPTCE	CSECT	RDO - terminal definition validation	OS	03
DFHSPTDB	CSECT	RDO - TDQueue definition validation	-	03
DFHSPTDC	CSECT	RDO - TDQueue definition validation	-	03
DFHSPTDE	CSECT	RDO - TDQueue definition validation	-	03
DFHSPTIB	CSECT	CSDUP - cross-keyword validation for sessions	OS	03
DFHSPTIC	CSECT	RDO - cross-keyword validation for sessions	OS	03
DFHSPTIE	CSECT	RDO - sessions definition validation	OS	03
DFHSPTNB	CSECT	CSDUP - cross-keyword validation for connections	OS	03
DFHSPTNE	CSECT	RDO - connection definition validation	OS	03
DFHSPTNC	CSECT	RDO - cross-keyword validation for connections	OS	03
DFHSPTRI	CSECT	SPI trace interpreter	OS	03
DFHSPTSB	CSECT		OS	03
DFHSPTSC	CSECT		OS	03
DFHSPTSE	CSECT		OS	03
DFHSPTYB	CSECT	CSDUP - cross-keyword validation for typeterms	OS	03
DFHSPTYC	CSECT	RDO - cross-keyword validation for typeterms	OS	03
DFHSPTYE	CSECT	RDO - Typeterms definition validation	OS	03
DFHSPP	CSECT	Syncpoint program	-	03
DFHSPXMB	CSECT	CSDUP - cross-keyword validation for transactions	-	03
DFHSPXMC	CSECT	RDO - cross-keyword validation for transactions	-	03
DFHSPXME	CSECT	RDO - TranClass definition validation	-	03
DFHSRADS	DSECT	SRB interface control area	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHSRASM	CSECT	Alias for SRRHASM	11	-
DFHSRCOB	CSECT	Alias for SRRCOBOL	C2	-
DFHSRED	DSECT	System recovery error data for XSRAB exit	11	-
DFHSRLI	CSECT	SRP LIFO storage subroutine	OS	03
DFHSRLIA	DSECT	SRLI parameter list	OS	-
DFHSRLIM	Macro	SRLI request	OS	-
DFHSRLIT	CSECT	SRLI trace interpretation data	OS	03
DFHSRP	CSECT	System recovery program	OS	03
DFHSRPLI	CSECT	Alias for SRRPLI	P2	-
DFHSRRC	CSECT	Alias for SRRC	-	08
DFHSRSRA	Source	SRSR parameter list	OS	-
DFHSRSRM	Source	SRSR request	OS	-
DFHSRT	Macro	System recovery table	11	-
DFHSRTDS	DSECT	System recovery table	OS	-
DFHSRT1$	Sample	System recovery table	19	03
DFHSRXDS	DSECT	SRB and extensions in SQA	OS	-
DFHSR1	CSECT	System recovery program	-	03
DFHSSAD	Macro	Static storage area address list	11	-
DFHSSDUF	CSECT (OCO)	SDUMP formatter for static storage areas	-	03
DFHSSEN	CSECT	Subsystem interface EOT and EOM routine	OS	03
DFHSSGC	CSECT	Subsystem interface generic connect	OS	03
DFHSSIN	CSECT	CICS subsystem initialization	OS	03
DFHSSMGP	CSECT	Subsystem interface message program	OS	03
DFHSSMGT	CSECT	Subsystem interface message table	OS	03
DFHSSREQ	Macro	Subsystem interface (SSI) request	OS	-
DFHSSWT	CSECT	Subsystem interface WTO router	OS	03
DFHSSWTF	CSECT	SSI MODIFY command password suppression	OS	03
DFHSSWTO	CSECT	SSI CICS console message reformatting	OS	03
DFHSTAB	Macro	Table scan macro	11	-
DFHSTACK	Macro	Save/restore registers on subroutine calls	OS	-
DFHSTART	Other	CICS startup cataloged procedure	02	-
DFHSTDBX	CSECT (OCO)	STUP - DBCTL statistics summary formatter	-	03
DFHSTDM	CSECT (OCO)	ST domain - initialization/termination	-	03
DFHSTDSX	CSECT (OCO)	STUP - DS domain stats summary formatter	-	03
DFHSTDUF	CSECT (OCO)	SDUMP formatter for ST domain	-	03
DFHSTDUX	CSECT (OCO)	STUP - DU domain stats summary formatter	-	03
DFHSTD2	Macro	Standard names of domains, gates, formats	11	-
DFHSTD2X	CSECT		-	03
DFHSTEJX	CSECT	Stats.Util.EJ Domain Extended formatting	-	03
DFHSTE15	CSECT (OCO)	STUP - DFSORT interface to E15 user exit	-	03
DFHSTE35	CSECT (OCO)	STUP - DFSORT interface to E35 user exit	-	03
DFHSTFC	CSECT	AP domain - file control statistics	-	03
DFHSTGDS	DSECT	ST domain - global statistics	11	-
DFHSTGDS	DSECT	ST domain - global statistics	C2	07
DFHSTIDS	DSECT	Statistics common record header and record identifiers	11	-
DFHSTIDS	DSECT	Statistics common record header and record identifiers	C2	07
DFHSTIIX	CSECT	Stats.Util.II Domain Extended formatting	-	03
DFHSTIN	CSECT (OCO)	STUP - DFSORT E15 user exit input routine	-	03
DFHSTLDX	CSECT (OCO)	STUP - LD domain stats summary formatter	-	03
DFHSTLGX	CSECT (OCO)	Logger Domain statistics extended	-	03
DFHSTLK	CSECT	AP domain - ISC/IRC statistics	-	03
DFHSTLS	CSECT	AP domain - LSR pool statistics	-	03
DFHSTMNX	CSECT (OCO)	STUP - MN domain stats summary formatter	-	03
DFHSTNDD	Macro		11	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHSTNQX	CSECT (OCO)	Enqueue Manager domain statistics	-	03
DFHSTOT	CSECT (OCO)	STUP - DFSORT E35 user exit output routine	-	03
DFHSTP	CSECT	System termination program	OS	03
DFHSTPGX	CSECT	STUP - PG domain autoinstall statistics	-	03
DFHSTRD	CSECT (OCO)	STUP - read interface	-	03
DFHSTRDA	DSECT	STRD parameter list	OS	-
DFHSTRDM	Macro	STRD request	OS	-
DFHSTRMX	CSECT (OCO)	Recovery Manager domain statistics	-	03
DFHSTSJX	CSECT	Stats.Util.JVM Domain Extended formatting	-	03
DFHSTSMF	Macro	ST domain - statistics SMF header and SMF product section	11	-
DFHSTSMX	CSECT (OCO)	STUP - SM domain stats summary formatter	-	03
DFHSTSOX	CSECT	Stats.Util.SO Domain Extended formatting	-	03
DFHSTST	CSECT (OCO)	ST domain - services	-	03
DFHSTSTA	DSECT	STST parameter list	OS	-
DFHSTSTM	Macro	STST request	OS	-
DFHSTSTT	CSECT	STST trace interpretation data	OS	03
DFHSTSTX	CSECT (OCO)	STUP - ST domain stats summary formatter	-	03
DFHSTSZ	CSECT	AP domain - FEPI statistics	-	03
DFHSTTD	CSECT	AP domain - transient data statistics	-	03
DFHSTTI	CSECT (OCO)	ST domain - timer notify handler	-	03
DFHSTTM	CSECT	AP domain - table manager statistics	-	03
DFHSTTQX	CSECT	STUP - TDQueue id extended formatting	-	03
DFHSTTR	CSECT	AP domain - terminal statistics	-	03
DFHSTTRI	CSECT (OCO)	Trace interpreter for ST domain	-	03
DFHSTTSX	CSECT (OCO)	Shared TS statistics	-	03
DFHSTUDB	CSECT (OCO)	STUP - DBCTL statistics formatter	-	03
DFHSTUDE	CSECT (OCO)	STUP - DE domain statistics formatter	-	03
DFHSTUDS	CSECT (OCO)	STUP - DS domain statistics formatter	-	03
DFHSTUDU	CSECT (OCO)	STUP - DU domain statistics formatter	-	03
DFHSTUD2	CSECT (OCO)	STUP - DU domain statistics formatter	-	03
DFHSTUE	CSECT (OCO)	ST domain - user exit service	-	03
DFHSTUEJ	CSECT	STUP - EJ Domain formatting routine	-	03
DFHSTUII	CSECT	STUP - II Domain formatting routine	-	03
DFHSTULD	CSECT (OCO)	STUP - LD domain statistics formatter	-	03
DFHSTULG	CSECT (OCO)	STUP - Logger domain formatting routine	-	03
DFHSTUMN	CSECT (OCO)	STUP - MN domain statistics formatter	-	03
DFHSTUNQ	CSECT (OCO)	STUP - Enqueue manager domain statistics	-	03
DFHSTUPG	CSECT (OCO)	STUP - PG domain autoinstall statistics formatter	-	03
DFHSTUP1	CSECT (OCO)	STUP - preinitialize	-	03
DFHSTURM	CSECT (OCO)	STUP - Recovery manager domain statistics	-	03
DFHSTURS	CSECT (OCO)	STUP - US domain statistics formatter	-	03
DFHSTURX	CSECT (OCO)	STUP - US domain statistics summary formatter	-	03
DFHSTUSJ	CSECT	STUP - Scaleable JVM Domain formatting	-	03
DFHSTUSM	CSECT (OCO)	STUP - SM domain statistics formatter	-	03
DFHSTUSO	CSECT	STUP - Sockets Domain formatting routine	-	03
DFHSTUST	CSECT (OCO)	STUP - ST domain statistics formatter	-	03
DFHSTUTQ	CSECT (OCO)	STUP - Transient data statistics	-	03
DFHSTUTS	CSECT (OCO)	Shared TS statistics	-	03
DFHSTUXC	CSECT (OCO)	STUP - Transaction manager domain statistics	-	03
DFHSTUXM	CSECT (OCO)	STUP - XM domain statistics formatter	-	03
DFHSTU03	CSECT (OCO)	STUP - VTAM statistics formatter	-	03
DFHSTU04	CSECT (OCO)	STUP - autoinstall terminals statistics formatter	-	03
DFHSTU06	CSECT (OCO)	STUP - terminal statistics formatter	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHSTU08	CSECT (OCO)	STUP - LSRPOOL resource statistics formatter	-	03
DFHSTU09	CSECT (OCO)	STUP - LSRPOOL file statistics formatter	-	03
DFHSTU14	CSECT (OCO)	STUP - ISC/IRC statistics formatter	-	03
DFHSTU16	CSECT (OCO)	STUP - table manager statistics formatter	-	03
DFHSTU17	CSECT (OCO)	STUP - file control statistics formatter	-	03
DFHSTU21	CSECT (OCO)	STUP - ISC/IRC attach-time statistics formatter	-	03
DFHSTU22	CSECT (OCO)	STUP - FEPI statistics formatter	-	03
DFHSTWR	CSECT (OCO)	STUP - write interface	-	03
DFHSTWRA	DSECT	STWR parameter list	OS	-
DFHSTWRM	Macro	STWR request	OS	-
DFHSTXCX	CSECT (OCO)	STUP - Transaction manager domain extended formatting routine for TranClass Stats	-	03
DFHSTXLE	CSECT	Off-line Statistics Utility Program	-	03
DFHSTXMX	CSECT (OCO)	STUP - XM statistics extended formatter	-	03
DFHST03X	CSECT (OCO)	STUP - VTAM statistics summary formatter	-	03
DFHST04X	CSECT (OCO)	STUP - autoinstall terminals statistics summary formatter	-	03
DFHST06X	CSECT (OCO)	STUP - terminal stats summary formatter	-	03
DFHST08X	CSECT (OCO)	STUP - LSRPOOL resource statistics summary formatter	-	03
DFHST09X	CSECT (OCO)	STUP - LSRPOOL file statistics summary formatter	-	03
DFHST14X	CSECT (OCO)	STUP - ISC/IRC stats summary formatter	-	03
DFHST16X	CSECT (OCO)	STUP - table manager statistics summary formatter	-	03
DFHST17X	CSECT (OCO)	STUP - file control statistics summary formatter	-	03
DFHST21X	CSECT (OCO)	STUP - ISC/IRC attach-time statistics summary formatter	-	03
DFHST22X	CSECT (OCO)	STUP - FEPI statistics summary formatter	-	03
DFHSUDUF	CSECT (OCO)	SDUMP formatter for DU domain summary	-	03
DFHSUEX	CSECT	User exit handler subroutine	-	03
DFHSUEXA	DSECT	SUEX parameter list	OS	-
DFHSUEXM	Macro	SUEX request	OS	-
DFHSUEXT	CSECT	SUEX trace interpretation data	OS	03
DFHSUME	CSECT (OCO)	ME domain - produce and issue messages subroutine (used by ME and LM domains)	-	03
DFHSUMEA	DSECT	SUME parameter list	OS	-
DFHSUMEM	Macro	SUME request	OS	-
DFHSUMET	CSECT	SUME trace interpretation data	-	03
DFHSUNP	Other		OS	-
DFHSUSX	CSECT	XRF signon	OS	03
DFHSUSXA	DSECT	SUSX parameter list	OS	-
DFHSUSXM	Macro	SUSX request	OS	-
DFHSUSXT	DSECT	SUSX translate tables	OS	03
DFHSUTRI	CSECT	WTO/WTOR subroutine trace interpreter	OS	03
DFHSUWT	CSECT	WTO/WTOR interface subroutine	OS	03
DFHSUWTA	DSECT	SUWT parameter list	OS	-
DFHSUWTM	Macro	SUWT request	OS	-
DFHSUWTT	CSECT	SUWT trace interpretation data	OS	03
DFHSUZX	CSECT	ZC trace controller	OS	03
DFHSUZXA	DSECT	SUZX parameter list	OS	-
DFHSUZXM	Macro	SUZX request	OS	-
DFHSUZXT	CSECT	SUZX trace interpretation data	OS	03
DFHSVCHK	Macro	SVC level check	11	-
DFHSWXK	Macro	Switch execution key routine	OS	-
DFHSYS	Macro	System definition macro	11	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHSZAPA	DSECT	FEPI programming copybook - assembler	11	-
DFHSZAPC	DSECT	FEPI programming copybook - C/370	-	08
DFHSZAPO	DSECT	FEPI programming copybook - COBOL	C2	-
DFHSZAPP	DSECT	FEPI programming copybook - PL/I	P2	17
DFHSZATC	CSECT (OCO)	FEPI adaptor command tables	-	03
DFHSZATR	CSECT (OCO)	FEPI adaptor program	-	03
DFHSZBCL	CSECT (OCO)	FEPI cleanup API requests at error routine	-	03
DFHSZBCS	CSECT (OCO)	FEPI RM collect statistics	-	03
DFHSZBFT	CSECT (OCO)	FEPI FREE transaction requests scheduler	-	03
DFHSZBLO	CSECT (OCO)	FEPI lost session reporter	-	03
DFHSZBRS	CSECT (OCO)	FEPI RM collect resource ID statistics	-	03
DFHSZBSI	CSECT (OCO)	FEPI signon exit scheduler	-	03
DFHSZBST	CSECT (OCO)	FEPI STSN transaction scheduler	-	03
DFHSZBUN	CSECT (OCO)	FEPI unsolicited data transaction scheduler	-	03
DFHSZBUS	CSECT (OCO)	FEPI RM unsolicited statistics recording	-	03
DFHSZDUF	CSECT (OCO)	FEPI dump formatting routine	-	03
DFHSZFRD	CSECT (OCO)	FEPI formatted 3270 RECEIVE support	-	03
DFHSZFSD	CSECT (OCO)	FEPI formatted 3270 SEND support	-	03
DFHSZIDX	CSECT (OCO)	FEPI SLU P queue install/discard exit	-	03
DFHSZPCP	CSECT (OCO)	FEPI SLU P flow controller	-	03
DFHSZPDX	CSECT (OCO)	FEPI SLU P drain completion exit	-	03
DFHSZPID	CSECT (OCO)	FEPI SLU P send data processor	-	03
DFHSZPIX	CSECT (OCO)	FEPI SLU P send completion exit	-	03
DFHSZPOA	CSECT (OCO)	FEPI SLU P send response processor	-	03
DFHSZPOD	CSECT (OCO)	FEPI SLU P receive data processor	-	03
DFHSZPOR	CSECT (OCO)	FEPI SLU P response processor	-	03
DFHSZPOX	CSECT (OCO)	FEPI SLU P receive specific response exit	-	03
DFHSZPOY	CSECT (OCO)	FEPI SLU P receive specific response processor	-	03
DFHSZPQS	CSECT (OCO)	FEPI SLU P REQSESS (request session) issuer	-	03
DFHSZPQX	CSECT (OCO)	FEPI SLU P REQSESS exit	-	03
DFHSZPSB	CSECT (OCO)	FEPI SLU P bind processor	-	03
DFHSZPSC	CSECT (OCO)	FEPI SLU P session controller	-	03
DFHSZPSD	CSECT (OCO)	FEPI SLU P SDT processor	-	03
DFHSZPSH	CSECT (OCO)	FEPI SLU P SHUTC processor	-	03
DFHSZPSQ	CSECT (OCO)	FEPI SLU P quiesce complete (QC) processor	-	03
DFHSZPSR	CSECT (OCO)	FEPI RESETSR processor CSECT	-	03
DFHSZPSS	CSECT (OCO)	FEPI SLU P STSN processor	-	03
DFHSZPSX	CSECT (OCO)	FEPI SLU P OPNSEC completion exit	-	03
DFHSZPTE	CSECT (OCO)	FEPI SLU P TERMSESS processor	-	03
DFHSZRCA	CSECT (OCO)	FEPI node control processor	-	03
DFHSZRCT	CSECT (OCO)	FEPI issue processor	-	03
DFHSZRDC	CSECT (OCO)	FEPI delete connection processor	-	03
DFHSZRDG	CSECT (OCO)	FEPI discard node processor	-	03
DFHSZRDN	CSECT (OCO)	FEPI delete node processor	-	03
DFHSZRDP	CSECT (OCO)	FEPI dispatcher	-	03
DFHSZRDS	CSECT (OCO)	FEPI discard property set processor	-	03
DFHSZRDT	CSECT (OCO)	FEPI discard target processor	-	03
DFHSZREQ	CSECT (OCO)	FEPI request processor	-	03
DFHSZRFC	CSECT (OCO)	FEPI FREE completion processor	-	03
DFHSZRGR	CSECT (OCO)	FEPI Dispatcher work queue processor	-	03
DFHSZRIA	CSECT (OCO)	FEPI allocate processor	-	03
DFHSZRIC	CSECT (OCO)	FEPI define connection processor	-	03
DFHSZRID	CSECT (OCO)	FEPI discard processor	-	03
DFHSZRIF	CSECT (OCO)	FEPI install free processor	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHSZRII	CSECT (OCO)	FEPI install processor	-	03
DFHSZRIN	CSECT (OCO)	FEPI install node processor	-	03
DFHSZRIO	CSECT (OCO)	FEPI ACB open processor	-	03
DFHSZRIP	CSECT (OCO)	FEPI install pool processor	-	03
DFHSZRIQ	CSECT (OCO)	FEPI inquire processor	-	03
DFHSZRIS	CSECT (OCO)	FEPI install processor	-	03
DFHSZRIT	CSECT (OCO)	FEPI install target processor	-	03
DFHSZRIW	CSECT (OCO)	FEPI SET processor	-	03
DFHSZRNC	CSECT (OCO)	FEPI NODE processor	-	03
DFHSZRNO	CSECT (OCO)	FEPI NOOP processor	-	03
DFHSZRPM	CSECT (OCO)	FEPI timer services	-	03
DFHSZRPW	CSECT (OCO)	FEPI request preparation	-	03
DFHSZRQR	CSECT (OCO)	FEPI queue for REQSESS processing	-	03
DFHSZRQW	CSECT (OCO)	FEPI request queue processor	-	03
DFHSZRRD	CSECT (OCO)	FEPI RECEIVE request processor	-	03
DFHSZRRT	CSECT (OCO)	FEPI request release processor	-	03
DFHSZRSC	CSECT (OCO)	FEPI connection processor	-	03
DFHSZRSE	CSECT (OCO)	FEPI SEND request processor	-	03
DFHSZRST	CSECT (OCO)	FEPI START request processor	-	03
DFHSZRTM	CSECT (OCO)	FEPI recovery services	-	03
DFHSZRXD	CSECT (OCO)	FEPI EXTRACT processor	-	03
DFHSZRZZ	CSECT (OCO)	FEPI TERMINATE processor	-	03
DFHSZSDS	DSECT	FEPI storage control block	11	-
DFHSZSIP	CSECT (OCO)	FEPI initialization processor	-	03
DFHSZVBN	CSECT (OCO)	FEPI copy NIB mask to real NIB	-	03
DFHSZVGF	CSECT (OCO)	FEPI get queue element FIFO	-	03
DFHSZVQS	CSECT (OCO)	FEPI REQSESS dispatcher	-	03
DFHSZVRA	CSECT (OCO)	FEPI VTAM receive_any processor	-	03
DFHSZVRI	CSECT (OCO)	FEPI VTAM receive_any issuer	-	03
DFHSZVSC	CSECT (OCO)	FEPI delayed bind processor	-	03
DFHSZVSL	CSECT (OCO)	FEPI SETLOGON request issuer	-	03
DFHSZVSQ	CSECT (OCO)	FEPI VTAM feedback interpreter	-	03
DFHSZVSR	CSECT (OCO)	FEPI VTAM feedback interpreter	-	03
DFHSZVSY	CSECT (OCO)	FEPI VTAM feedback interpreter	-	03
DFHSZWSL	CSECT (OCO)	FEPI RPL exit after SETLOGON	-	03
DFHSZXDA	CSECT (OCO)	FEPI VTAM DFASY exit	-	03
DFHSZXFR	CSECT (OCO)	FEPI RPL exit to free request block	-	03
DFHSZXLG	CSECT (OCO)	FEPI VTAM logon exit	-	03
DFHSZXLT	CSECT (OCO)	FEPI VTAM LOSTERM (lost terminal) exit	-	03
DFHSZXNS	CSECT (OCO)	FEPI VTAM NSEXIT (network services) exit	-	03
DFHSZXPM	CSECT (OCO)	FEPI STIMER IRB exit routine	-	03
DFHSZXRA	CSECT (OCO)	FEPI VTAM RECEIVE_ANY exit	-	03
DFHSZXSC	CSECT (OCO)	FEPI VTAM SCIP (session control) exit	-	03
DFHSZXTP	CSECT (OCO)	FEPI VTAM TPEND exit	-	03
DFHSZYLG	CSECT (OCO)	FEPI RPL exit following logon reject	-	03
DFHSZYQR	CSECT (OCO)	FEPI post for REQSESS processing	-	03
DFHSZYRI	CSECT (OCO)	FEPI VTAM RECEIVE_ANY issuer	-	03
DFHSZYSC	CSECT (OCO)	FEPI VTAM SCIP exit extension	-	03
DFHSZYSR	CSECT (OCO)	FEPI VTAM feedback interpreter	-	03
DFHSZYSY	CSECT (OCO)	FEPI VTAM feedback interpreter	-	03
DFHSZZAG	CSECT (OCO)	FEPI get RECEIVE_ANY request block	-	03
DFHSZZFR	CSECT (OCO)	FEPI free RECEIVE_ANY request block	-	03
DFHSZZNG	CSECT (OCO)	FEPI get session control request block	-	03
DFHSZZRG	CSECT (OCO)	FEPI get RPL request block	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHSZ2CP	CSECT (OCO)	FEPI SLU2 flow controller	-	03
DFHSZ2DX	CSECT (OCO)	FEPI SLU2 drain completion exit	-	03
DFHSZ2ID	CSECT (OCO)	FEPI SLU2 send data processor	-	03
DFHSZ2IX	CSECT (OCO)	FEPI SLU2 send completion exit	-	03
DFHSZ2OA	CSECT (OCO)	FEPI SLU2 send response processor	-	03
DFHSZ2OD	CSECT (OCO)	FEPI SLU2 receive data processor	-	03
DFHSZ2OR	CSECT (OCO)	FEPI SLU2 response processor	-	03
DFHSZ2OX	CSECT (OCO)	FEPI SLU2 receive specific completion exit	-	03
DFHSZ2OY	CSECT (OCO)	FEPI SLU2 receive specific action module	-	03
DFHSZ2PX	CSECT (OCO)	FEPI SLU2 positive response drain exit	-	03
DFHSZ2QS	CSECT (OCO)	FEPI SLU2 REQSESS issuer	-	03
DFHSZ2QX	CSECT (OCO)	FEPI SLU2 REQSESS exit	-	03
DFHSZ2SB	CSECT (OCO)	FEPI SLU2 bind processor	-	03
DFHSZ2SC	CSECT (OCO)	FEPI SLU2 session controller	-	03
DFHSZ2SD	CSECT (OCO)	FEPI SLU2 SDT processor	-	03
DFHSZ2SH	CSECT (OCO)	FEPI SLU2 SHUTC processor	-	03
DFHSZ2SQ	CSECT (OCO)	FEPI SLU2 QC processor	-	03
DFHSZ2SR	CSECT (OCO)	FEPI SLU2 RESETSR processor	-	03
DFHSZ2SX	CSECT (OCO)	FEPI SLU2 OPNSEC processor	-	03
DFHSZ2TE	CSECT (OCO)	FEPI SLU2 TERMSESS processor	-	03
DFHTACB	Macro	Task abend control block	11	-
DFHTACLE	DSECT	TCT line entry prefix	11	-
DFHTACP	CSECT	Terminal abnormal condition program	OS	03
DFHTAJP	CSECT	Time adjustment program	OS	03
DFHTBS	Macro	Builder interface	OS	-
DFHTBSB	CSECT	Add a node	OS	03
DFHTBSBP	CSECT	Recursive part of DFHTBSB	OS	03
DFHTBSD	CSECT	Delete node program	OS	03
DFHTBSDP	CSECT	Recursive part of DFHTBSD	OS	03
DFHTBSL	CSECT	Create recovery record for node	OS	03
DFHTBSLP	CSECT	Recursive part of DFHTBSL	OS	03
DFHTBSQ	CSECT	Builder inquire process	OS	03
DFHTBSQP	CSECT	Recursive part of DFHTBSQ	OS	03
DFHTBSR	CSECT	Builder restore process	OS	03
DFHTBSRP	CSECT	Recursive part of DFHTBSR	OS	03
DFHTBSS	CSECT	TBS syncpoint processor	-	03
DFHTBSST	DSECT	TBSS translate tables	-	03
DFHTBS00	CSECT	Table builder services program	OS	03
DFHTC	Macro	Terminal service request	11	-
DFHTCA	Macro	Task control area	11	-
DFHTCADS	DSECT	Task control area	11	-
DFHTCAM	Source	CICS-TCAM interface logic	OS	-
DFHTCCLC	Source	Common line control logic	OS	-
DFHTCCOM	Source	Input data length computation	OS	-
DFHTCCSS	Source	Start-stop event analysis	OS	-
DFHTCDEF	Symbolic	Terminal control definitions	OS	-
DFHTCDPF	CSECT (OCO)	Terminal control prefix SDUMP module	-	03
DFHTCDUF	CSECT (OCO)	Terminal control SDUMP formatter	-	03
DFHTCORS	Source	Terminal storage routine	OS	-
DFHTCP	CSECT	Terminal control program	OS	03
DFHTCPCL	Macro	DFHZCP request	OS	-
DFHTCPCM	Macro	Common ZCP functions	11	-
DFHTCPLR	Macro	LU6.2 limited resources service	OS	-
DFHTCPQR	Macro	Queued response notification	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHTCPRA	DSECT	Receive-any control element	OS	-
DFHTCPRT	Macro	DFHZCP RETURN macro	OS	-
DFHTCPSM	Macro	TCT generation - VTAM DSECTs	11	-
DFHTCPSV	Macro	DFHZCP SAVE macro	OS	-
DFHTCPZR	Macro	VTAM RPL extension for HPO	11	-
DFHTCQUE	Macro	DFHZCP QUEUE macro	OS	-
DFHTCRP	CSECT	Terminal control recovery program	OS	03
DFHTCRPC	CSECT	XRF tracking interface for TCT contents	OS	03
DFHTCRPL	CSECT	Install TCT macro definitions	OS	03
DFHTCRPS	CSECT	XRF tracking interface for ZCP sessions	OS	03
DFHTCRPU	CSECT	XRF tracking interface for SNTTEs	OS	03
DFHTCRWE	DSECT	Remote install work element	OS	-
DFHTCSAM	Source	Sequential terminal logic	OS	-
DFHTCSRV	Macro	DFHTC inner service macro	11	-
DFHTCSUM	CSECT	Terminal control dump summary program	-	03
DFHTCT	Macro	Terminal control table	11	-
DFHTCTDY	CSECT	Terminal control table (dummy)	19	03
DFHTCTFN	Source	TCT TYPE=FINAL (VTAM)	11	-
DFHTCTFX	DSECT	TCT prefix	11	-
DFHTCTI	Source	Terminal control task initiation logic	OS	-
DFHTCTLC	Macro	TCT inner macro	11	-
DFHTCTLE	DSECT	TCT line entry	11	-
DFHTCTME	Macro	Generate TCT mode group entries	11	-
DFHTCTPR	Macro	TCTTE partition extension builder	11	-
DFHTCTPS	Macro	TCT inner macro	11	-
DFHTCTPX	Macro	TCT inner macro	11	-
DFHTCTRD	Macro	VTAM RDO command list builder	11	-
DFHTCTRE	Macro	TCT definition macro	11	-
DFHTCTRN	Source	Terminal control translation tables	OS	-
DFHTCTSA	Macro	TCT inner macro	11	-
DFHTCTSB	Macro	TCT inner macro	11	-
DFHTCTSE	Macro	Generate ISC system entry	11	-
DFHTCTSK	Macro	Generate TCT skeleton entry	11	-
DFHTCTST	Macro	TCT inner macro	11	-
DFHTCTSV	Macro	TCT inner macro	11	-
DFHTCTTE	DSECT	TCT terminal entry	11	-
DFHTCTUA	Macro	TCT inner macro	11	-
DFHTCTUB	Macro	TCT inner macro	11	-
DFHTCTWA	DSECT	TC transaction work area	11	-
DFHTCTWE	DSECT	TCT autodefine work element	OS	-
DFHTCTZE	Macro	TCTTE definition	11	-
DFHTCT5$	Sample	Terminal control table	19	03
DFHTCUDS	DSECT	COMMAREA passed to autoinstall exit	11	-
DFHTCUDS	DSECT	COMMAREA passed to autoinstall exit	C2	07
DFHTCUDS	DSECT	COMMAREA passed to autoinstall exit	P2	08
DFHTCV29	DSECT	XRF session state data control vector	OS	-
DFHTCX	Macro	TCA extension for LU6.2	11	-
DFHTCXDF	CSECT	DU domain - transaction dump formatter for terminal related areas	OS	03
DFHTD	Macro	Transient data service request	11	-
DFHTDA	CSECT	Transient data request processor	-	03
DFHTDB	CSECT	Transient data request processor	-	03
DFHTDCI	DSECT	Transient data VSAM CI map	OS	-
DFHTDDUF	CSECT (OCO)	Transient data SDUMP formatter	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHTDEXL	CSECT	Transient data DCB exit list and DCB abend exit routine	OS	03
DFHTDGDS	DSECT	Transaction dump global statistics	11	-
DFHTDGDS	DSECT	Transaction dump global statistics	C2	07
DFHTDOA	DSECT	Transient data output area	11	-
DFHTDOC	CSECT	Transient data open/close for extrapartition queues	-	03
DFHTDOCA	DSECT	TDOC parameter list	OS	-
DFHTDOCM	Macro	TDOC request	OS	-
DFHTDOCT	CSECT	TDOC trace interpretation data	-	03
DFHTDRDS	DSECT	Transaction dump statistics by dump code	11	-
DFHTDRDS	DSECT	Transaction dump statistics by dump code	C2	07
DFHTDRDS	DSECT	Transaction dump statistics by dump code	P2	-
DFHTDRP	CSECT	Transient data recovery program	OS	03
DFHTDSDS	DSECT	Transient data static storage	OS	-
DFHTDTDA	DSECT	TDTD parameter list	OS	-
DFHTDTDM	Macro	TDTD request	OS	-
DFHTDTDT	CSECT	TDTD trace interpretation data	-	03
DFHTDTM	CSECT	Transient data table management gate	-	03
DFHTDTMA	CSECT	TDTM parameter list	OS	-
DFHTDTMM	Macro	TDTM request	OS	-
DFHTDTMT	DSECT	TDTM translate tables	-	03
DFHTDTRI	CSECT	Transient data trace interpreter	OS	03
DFHTDUED	Macro	TD user exits EXEC argument list	11	-
DFHTDX	CSECT	Transient data phase 1 initialization	OS	03
DFHTDXM	CSECT (OCO)	XM domain - TD facility management services	OS	03
DFHTDXMA	DSECT	TDXM parameter list	OS	-
DFHTDXMM	Macro	TDXM request	OS	-
DFHTDXMT	CSECT (OCO)	TDXM trace interpretation data	OS	03
DFHTEPA	Macro	TEP inner macro	11	-
DFHTEPC	Macro	TEP inner macro	11	-
DFHTEPCA	Macro	TEP communication area	11	-
DFHTEPM	Macro	TEP module generator	11	-
DFHTEPS	Macro	TEP inner macro	11	-
DFHTEPT	Macro	TEP table generator	11	-
DFHTERID	Symbolic	Terminal error definitions	11	-
DFHTEST	Macro	Domain call argument TEST macro	11	-
DFHTFALA	DSECT	TFAL parameter list	OS	-
DFHTFALM	Macro	TFAL request	OS	-
DFHTFALT	CSECT (OCO)	TFAL trace interpretation data	-	03
DFHTFBFA	DSECT	TFBF parameter list	OS	-
DFHTFBFM	Macro	TFBF request	OS	-
DFHTFBFT	CSECT (OCO)	TFBF trace interpretation data	-	03
DFHTFIQ	CSECT (OCO)	Terminal facility manager inquire/set functions	-	03
DFHTFIQA	DSECT	TFIQ parameter list	OS	-
DFHTFIQI	DSECT	TFIQ requests (inline form)	OS	-
DFHTFIQM	DSECT	TFIQ requests	OS	-
DFHTFIQT	CSECT (OCO)	TFIQ trace interpretation data	-	03
DFHTFP	CSECT	Transaction failure program	OS	03
DFHTFRF	CSECT (OCO)	Terminal facility manager release function	-	03
DFHTFRFT	CSECT (OCO)	TFRF trace interpretation data	-	03
DFHTFTRI	CSECT (OCO)	Terminal facility manager trace interpreter	-	03
DFHTFXM	CSECT	TF XM transaction attach	-	03
DFHTIDM	CSECT (OCO)	TI domain - initialization/termination	-	03
DFHTIDUF	CSECT (OCO)	SDUMP formatter for TI domain	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHTIEDS	DSECT	Task interface element	OS	-
DFHTIEM	CSECT	Resource manager interface TIE manager	OS	03
DFHTIOA	DSECT	Terminal input/output area	11	-
DFHTIOA	DSECT	Terminal input/output area	C2	07
DFHTISR	CSECT (OCO)	TI domain - services	-	03
DFHTISRA	DSECT	TISR parameter list	OS	-
DFHTISRM	Macro	TISR request	OS	-
DFHTISRT	CSECT	TISR trace interpretation data	-	03
DFHTITRI	CSECT (OCO)	Trace interpreter for TI domain	-	03
DFHTLT	Macro	Terminal list table	11	-
DFHTM	Macro	Table manager interface	11	-
DFHTMDUF	CSECT (OCO)	Table manager SDUMP formatter	-	03
DFHTMP01	CSECT (OCO)	Table manager program - part 1	-	03
DFHTMP02	CSECT (OCO)	Table manager program - part 2	-	03
DFHTMTRI	CSECT (OCO)	Table manager program trace interpreter	-	03
DFHTOACN	CSECT	Terminal object resolution (TOR) - add connection	OS	03
DFHTOAPT	CSECT	TOR - add pooled terminal	-	03
DFHTOASE	CSECT	TOR - add session	OS	03
DFHTOATM	CSECT	TOR - add (non-pooled) terminal	-	03
DFHTOATY	CSECT	TOR - add typeterm	-	03
DFHTOBPS	CSECT	TOR - create BPS and check attributes	OS	03
DFHTOCAN	CSECT	TOR - dynamic backout processing	-	03
DFHTOCMT	CSECT	TOR - syncpoint commit processing	-	03
DFHTOLCR	CSECT	TOR - end logical unit of complex replacement	-	03
DFHTOLUI	CSECT	TOR - end logical unit of installation	-	03
DFHTOM	Macro	BMS terminal output	OS	-
DFHTON	CSECT	Terminal object resolution module	-	03
DFHTONR	CSECT	Terminal object resolution recovery	-	03
DFHTONRT	DSECT	TONR translate tables	-	03
DFHTORP	CSECT	Terminal object recovery program	-	03
DFHTOR00	CSECT	Terminal object resolution program (DFHTOR)	OS	03
DFHTOUT1	CSECT	TOR - set operation utilities	-	03
DFHTOUT2	CSECT	TOR - map operation utilities	-	03
DFHTPE	DSECT	Terminal partition extension	OS	-
DFHTPP	CSECT	BMS terminal page processor	OS	-
DFHTPPA$	CSECT	BMS terminal page processor (standard)	OS	03
DFHTPP1$	CSECT	BMS terminal page processor (full)	OS	03
DFHTPQ	CSECT	BMS terminal page cleanup program	OS	03
DFHTQGDS	CSECT	Global statistics for Transient Data	11	-
DFHTQGDS	CSECT	Global statistics for Transient Data	C2	07
DFHTQRDS	CSECT	Transient data queue statistics	11	-
DFHTQRDS	CSECT	Transient data queue statistics	C2	07
DFHTPR	CSECT	BMS terminal page retrieval program	OS	03
DFHTPS	CSECT	BMS terminal page scheduling program	OS	03
DFHTR	Macro	Trace service request	11	-
DFHTRA	DSECT	TR domain - anchor block	OS	-
DFHTRACE	Macro	Trace system macro	OS	-
DFHTRADS	DSECT	TR domain - parameter list to DFHTRAP	11	-
DFHTRAO	CSECT	TR domain - auxiliary trace output	OS	03
DFHTRAP	CSECT	TR domain - FE global trap/trace exit	11	03
DFHTRBL	DSECT	TR domain - internal trace table block	OS	-
DFHTRCIF	CSECT	CZ Direct-to-CICS	-	03
DFHTRDM	CSECT	TR domain - initialization/termination	OS	03
DFHTRDS	DSECT	TR domain - control blocks	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHTRDUB	CSECT	TR and DU keyword copybook	OS	-
DFHTRDUF	CSECT (OCO)	SDUMP formatter for TR domain	-	03
DFHTREND	DSECT	TR domain - trace entry	11	-
DFHTREX	DSECT		-	03
DFHTRFCA	DSECT	Offline trace formatting control area	OS	-
DFHTRFFD	CSECT	Offline trace formatting - format data fields	OS	03
DFHTRFFE	CSECT	Offline trace formatting - format trace entry	OS	03
DFHTRFPB	CSECT	Offline trace formatting - process block	OS	03
DFHTRFPP	CSECT	Offline trace formatting - process selective print parameters	OS	03
DFHTRFT	CSECT	Trace put routine for features	OS	03
DFHTRFTA	CSECT	TRFT parameter list	OS	-
DFHTRFTD	CSECT	TR feature trace entry header	OS	-
DFHTRFTM	Macro	TRFT macro	OS	-
DFHTRFTT	CSECT	TRFT translate tables	OS	03
DFHTRFTX	Macro	TRFT macro	11	-
DFHTRFTY	Macro	TRFT call structured parameter list	11	-
DFHTRIB	CSECT	Trace interpretation string builder	OS	03
DFHTRP	CSECT	Trace control program	OS	03
DFHTRPRA	CSECT	Auxiliary trace offline formatting	OS	03
DFHTRPRG	CSECT	GTF trace offline formatting	OS	03
DFHTRPT	CSECT	TR domain - trace put (all destinations)	OS	03
DFHTRPTA	DSECT	TRPT parameter list	OS	-
DFHTRPTM	Macro	TRPT request	OS	-
DFHTRPTT	CSECT	TRPT trace interpretation data	OS	03
DFHTRPTX	Macro	TRPT request (XPI)	11	-
DFHTRPTY	DSECT	TRPT parameter list (XPI)	11	-
DFHTRPX	CSECT	TR domain - trace put (fast path)	OS	03
DFHTRSR	CSECT	TR domain - trace destination services	OS	03
DFHTRSRA	DSECT	TRSR parameter list	OS	-
DFHTRSRM	Macro	TRSR request	OS	-
DFHTRSRT	CSECT	TRSR trace interpretation data	OS	03
DFHTRSU	CSECT	TR domain - subroutines	OS	03
DFHTRSUA	DSECT	TRSU parameter list	OS	-
DFHTRSUM	Macro	TRSU request	OS	-
DFHTRSUT	CSECT	TRSU trace interpretation data	OS	03
DFHTRTRI	CSECT	Trace interpreter for TR domain	OS	03
DFHTRTST	Macro	TR domain - test if trace point active	OS	-
DFHTRUDS	DSECT	TRUE 24-bit parameter list save area	11	-
DFHTRXDF	CSECT	DU domain - transaction dump formatter for internal trace table	OS	03
DFHTRZCP	CSECT	Terminal object builder	OS	03
DFHTRZIP	CSECT	Session object builder	OS	03
DFHTRZPP	CSECT	Pool object builder	OS	03
DFHTRZXP	CSECT	Connection object builder	OS	03
DFHTRZYP	CSECT	Typeterm object builder	OS	03
DFHTRZZP	CSECT	Terminal object matching	OS	03
DFHTS	Macro	Temporary-storage service request	11	-
DFHTSAD	CSECT	TS Domain - TSAD Gate Function	-	03
DFHTSADT	CSECT		-	03
DFHTSAM	CSECT	TS auxiliary manager functions subroutine	-	03
DFHTSAMT	DSECT	TSAM translate tables	-	03
DFHTSBR	CSECT	TS browse functions	-	03
DFHTSBRA	CSECT	TSBR parameter list	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHTSBRM	Macro	TSBR request	OS	-
DFHTSBRT	DSECT	TSBR translate tables	-	03
DFHTSDM	CSECT	TS domain manager functions (initialize, quiesce, terminate)	-	03
DFHTSDQ	CSECT	Temporary Storage Delete Queue	-	03
DFHTSDUC	CSECT (OCO)	Temporary-storage SDUMP analysis	-	03
DFHTSDUF	CSECT (OCO)	Temporary-storage SDUMP formatter	-	03
DFHTSDUS	CSECT (OCO)	Temporary-storage SDUMP summary	-	03
DFHTSGDS	DSECT	Temporary-storage statistics DSECT (Assembler)	11	-
DFHTSGDS	DSECT	Temporary-storage statistics DSECT (COBOL)	C2	07
DFHTSHD	Macro	Temporary-storage input/output area header	OS	-
DFHTSIOA	DSECT	Temporary-storage input/output area	11	-
DFHTSICT	CSECT	TSIC translate tables	-	03
DFHTSITR	CSECT	TS trace interpretation	-	03
DFHTSMB	CSECT	DFHTSMB Design	-	03
DFHTSMBT	CSECT		-	03
DFHTSP	CSECT	Temporary-storage control program	OS	03
DFHTSPT	CSECT	TS put functions	-	03
DFHTSPTA	CSECT	TSPT request	OS	-
DFHTSPTM	Macro	TSPT request	OS	-
DFHTSPTT	DSECT	TSPT translate tables	-	03
DFHTSQR	CSECT	TS mainline queue request functions	-	03
DFHTSQRT	DSECT	TSQR translate tables	-	03
DFHTSRM	CSECT	TS recovery manager functions	-	03
DFHTSSBT	DSECT	TSSB translate tables	-	03
DFHTSSH	CSECT	TS shared TS functions	-	03
DFHTSSHT	DSECT	TSSH translate tables	-	03
DFHTSSR	CSECT	TS service functions (inquire, set)	-	03
DFHTSSRT	DSECT	TSSR translate tables	-	03
DFHTSST	CSECT	TS statistics functions	-	03
DFHTST	Macro	Temporary-storage table	11	-
DFHTSTDS	DSECT	Temporary-storage table	OS	-
DFHTSUED	CSECT	XTSEREQ and XTSEREQC EXEC parameter lists	11	-
DFHTSUTC	DSECT	TSUT abstract type internal control blocks	OS	-
DFHTSUTI	Macro	TSUT abstract type inline functions	OS	-
DFHTSWQ	CSECT	TS wait queue functions subroutine	-	03
DFHTSWQT	DSECT	TSWQ translate tables	-	03
DFHTTPDS	DSECT	BMS - terminal type parameter	11	-
DFHTUL	DSECT	Standard-labeled tape user labels	-	-
DFHTUTEN	Macro	Trace table generation macro	OS	-
DFHUCNV	Sample	CICS OS/2 user data conversion program	19	03
DFHUEDUF	CSECT (OCO)	User exit SDUMP formatter	-	03
DFHUEFDS	DSECT	File control user exit file/data set info	11	-
DFHUEH	CSECT	User exit handler (AP domain)	-	03
DFHUEHC	Source	User exit program invocation	-	-
DFHUEHWA	DSECT	User exit work areas	OS	-
DFHUEIQ	CSECT	User exit inquire exitprogram function	-	03
DFHUEIQT	CSECT	EIQT trace interpreter	-	03
DFHUEM	CSECT	User exit manager	OS	03
DFHUEPBD	DSECT	User exit program block	11	-
DFHUEPLD	DSECT	User exit program link	11	-
DFHUERMD	DSECT	User exit resource manager	11	-
DFHUETED	DSECT	User exit table entry	11	-
DFHUETHD	DSECT	User exit table header	11	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHUEXIT	Macro	User-exit-dependent code generator	11	-
DFHUEXPT	Macro	User exit point definition	11	-
DFHUIBA	DSECT	Assembler DSECT for User interface block	11	-
DFHUIBC	CSECT	C structure of the UIB	-	08
DFHUIBO	CSECT	Cobol structure of the UIB	C2	07
DFHUIBP	CSECT	PLI structure of the UIB	P2	17
DFHUPDVS	Other	Cataloged procedure to update a temporary library during system generation	18	-
DFHURLDS	DSECT	BMS - user-supplied route list	11	-
DFHURLDS	DSECT	BMS - user-supplied route list	C2	07
DFHURLDS	DSECT	BMS - user-supplied route list	D2	08
DFHUSAD	CSECT (OCO)	US domain - Add, Delete and Inquire User	-	03
DFHUSADA	DSECT	USAD parameter list	OS	-
DFHUSADM	Macro	USAD request	OS	-
DFHUSADT	CSECT (OCO)	USAD trace interpretation data	-	03
DFHUSAGE	Macro	Usage pricing code generation macro	OS	-
DFHUSAND	CSECT (OCO)	US domain - anchor block	OS	-
DFHUSBP	CSECT	User backout program	OS	03
DFHUSDET	DSECT	USDE translate tables	-	03
DFHUSDM	CSECT (OCO)	US domain - initialize, quiesce, and terminate domain functions	-	03
DFHUSDUF	CSECT (OCO)	US domain - dump formatter	-	03
DFHUSFL	CSECT (OCO)	US domain - Flatten and unflatten user	-	03
DFHUSFLA	DSECT	USFL parameter list	OS	-
DFHUSFLM	Macro	USFL request	OS	-
DFHUSFLT	CSECT (OCO)	USFL trace interpretation data	-	03
DFHUSGDS	DSECT	US domain - global statistics	11	-
DFHUSGDS	DSECT	US domain - global statistics	C2	07
DFHUSIS	CSECT (OCO)	US domain - inquire and set functions	-	03
DFHUSISA	DSECT	USIS parameter list	OS	-
DFHUSISM	Macro	USIS request	OS	-
DFHUSIST	CSECT (OCO)	USIS trace interpretation data	-	03
DFHUSST	CSECT (OCO)	US domain - statistics	-	03
DFHUSTI	CSECT (OCO)	US domain - timeout handler	-	03
DFHUSTIA	DSECT	USTI parameter list	OS	-
DFHUSTIM	Macro	USTI request	OS	-
DFHUSTIT	CSECT (OCO)	USTI trace interpretation data	-	03
DFHUSTRI	CSECT (OCO)	US domain - trace formatter	-	03
DFHUSXM	CSECT (OCO)	US domain - transaction support	-	03
DFHUSXMA	DSECT	USXM parameter list	OS	-
DFHUSXMI	Macro	USXM request (inline version of DFHUSXMM)	OS	-
DFHUSXMM	Macro	USXM request	OS	-
DFHUSXMT	CSECT (OCO)	USXM trace interpretation data	-	03
DFHUT64	CSECT	RU Base64 encoding and decoding	-	03
DFHVM	Macro	Version/modification level generator	11	-
DFHVSWA	DSECT	VSAM work area	11	-
DFHVTWA	DSECT	NACP LIFO storage definition	OS	-
DFHWBA	CSECT	Web module	-	03
DFHWBADX	CSECT	Web module	19	03
DFHWBAHX	CSECT	Web module	-	19
DFHWBALX	CSECT	Web module	-	19
DFHWBAOX	CSECT	Web module	-	19
DFHWBAP	CSECT	WB Domain WBAP Gate Functions	-	03
DFHWBAPF	CSECT	Web Module	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHWBAPT	CSECT	Web Module	-	03
DFHWBAP@	CSECT	Web module	-	03
DFHWBA1	CSECT	Web module	-	03
DFHWBA1D	CSECT	Web module	11	-
DFHWBA1H	CSECT	Web module	-	08
DFHWBA1L	CSECT	Web module	-	17
DFHWBA1O	CSECT	Web module	-	07
DFHWBBLI	CSECT	Business Logic interfac program	-	03
DFHWBBLL	CSECT		-	17
DFHWBBMS	CSECT	WB Web Interface BMS Support	-	03
DFHWBCDD	CSECT	Web module	-	11
DFHWBCDH	CSECT	Web module	-	08
DFHWBCDL	CSECT	Web module	-	17
DFHWBCDO	CSECT	Web module	-	07
DFHWBCNV	Macro	WB CICS Web Interface codepage macro	11	-
DFHWBC01	CSECT	Web module	-	03
DFHWBDCD	CSECT	Web module	OS	-
DFHWBDL@	CSECT	Autocall SCEEOBJ	-	03
DFHWBDM	CSECT	Domain initialization	-	03
DFHWBDUF	CSECT	Web module	-	03
DFHWBENV	CSECT	Web module	-	03
DFHWBEP	CSECT	Web error program	-	03
DFHWBEPL	CSECT		-	17
DFHWBGB	CSECT	WB Web Interface Garbage Collection	-	03
DFHWBIMG	CSECT	Web module	-	03
DFHWBIP	CSECT	Web module	-	03
DFHWBIPA	CSECT	Web module	OS	-
DFHWBIPM	Macro	DFHWBIP interface macro	11	-
DFHWBIPT	CSECT	Web module	-	03
DFHWBLT	CSECT	Web module	-	03
DFHWBOUT	CSECT	Web module	11	-
DFHWBPA	CSECT	Web module	-	03
DFHWBQM	CSECT	Domain Initialization	-	03
DFHWBQMT	CSECT		-	03
DFHWBRP	CSECT	Web module	-	03
DFHWBSR	CSECT	WB Web Send/Receive	-	03
DFHWBSRT	CSECT		-	03
DFHWBST	CSECT	Web module	-	03
DFHWBSTT	CSECT	Web module	-	03
DFHWBTC	CSECT	Web module	-	03
DFHWBTC@	CSECT	Web module	-	03
DFHWBTCT	CSECT	Web module	-	03
DFHWBTDD	CSECT	Web module	11	-
DFHWBTDH	CSECT	Web module	-	08
DFHWBTDL	CSECT	Web module	-	17
DFHWBTDO	CSECT	Web module	-	07
DFHWBTL	CSECT	Web module	-	03
DFHWBTLD	CSECT	Web module	11	-
DFHWBTLG	CSECT	Web module	11	-
DFHWBTLH	CSECT	Web module	-	08
DFHWBTLL	CSECT	Web module	-	17
DFHWBTLO	CSECT	Web module	C2	-
DFHWBTRI	CSECT	Web module	-	03
DFHWBTR1	CSECT	Web GWAPI Trace Interpretation	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHWBTRU	CSECT	Web module	-	03
DFHWBTTA	CSECT	Web module	-	03
DFHWBUCD	CSECT	Web module	11	-
DFHWBUCH	CSECT	Web module	-	08
DFHWBUCL	CSECT	Web module	-	17
DFHWBUCO	CSECT	Web module	-	07
DFHWBUN	CSECT	Web Interface Unescaping Program	-	03
DFHWBUND	CSECT	Web Interface Unescaping parameter list	11	-
DFHWBUNH	CSECT		-	08
DFHWBUNL	CSECT		-	17
DFHWBUNO	CSECT		-	07
DFHWBXM	CSECT	Web Interface Attach Client	-	03
DFHWBXMT	CSECT		-	03
DFHWBXN	CSECT	Web Attach Processing	-	03
DFHWCCS	CSECT	CAVM common services	OS	03
DFHWCGDS	DSECT	CAVM global control block	OS	-
DFHWCGNT	CSECT	CAVM entry point table for routines above 16MB line	OS	03
DFHWCSDS	DSECT	XRF static storage	OS	-
DFHWDATT	CSECT	XRF process dispatcher attach control	OS	03
DFHWDINA	CSECT	XRF process dispatcher initialization	OS	03
DFHWDISP	CSECT	XRF process dispatcher	OS	03
DFHWDSDS	DSECT	CAVM dispatcher interface parameter block	OS	-
DFHWDSRP	CSECT	PC/ABEND handler for XRF dispatcher	OS	03
DFHWDWAT	CSECT	XRF process dispatcher wait services	OS	03
DFHWFGDS	DSECT	CAVM file control block	OS	-
DFHWKP	CSECT	Warm keypoint program	-	03
DFHWLF	Macro	XRF LIFO free storage request	OS	-
DFHWLFRE	CSECT	XRF LIFO free allocation service	OS	03
DFHWLG	Macro	XRF LIFO get storage request	OS	-
DFHWLGET	CSECT	XRF LIFO get allocation service	OS	03
DFHWLIST	CSECT	WORDLIST function (used by DFHDBME)	OS	03
DFHWMG1	CSECT	XRF message manager, GETMSG process	OS	03
DFHWMI	CSECT	XRF message manager, signon initialization routine	OS	03
DFHWMMT	CSECT	XRF message manager, I/O services	OS	03
DFHWMPG	CSECT	XRF message manager, data copying service	OS	03
DFHWMP1	CSECT	XRF message manager, PUTMSG process	OS	03
DFHWMQG	CSECT	XRF message manager, CICS TCB part of GETMSG processing	OS	03
DFHWMQH	CSECT	XRF message manager, message block services for GETMSG	OS	03
DFHWMQP	CSECT	XRF message manager, CICS TCB part of PUTMSG processing	OS	03
DFHWMQS	CSECT	XRF message manager, work queue services	OS	03
DFHWMRD	CSECT	XRF message manager, message reader	OS	03
DFHWMS	CSECT	XRF message manager, request interface	OS	03
DFHWMS20	CSECT	XRF message manager, request router	OS	03
DFHWMWR	CSECT	XRF message manager, output routine	OS	03
DFHWNFDS	DSECT	CAVM NOTIFY exit parameter block	OS	-
DFHWORDS	CSECT	WORDS function (used by DFHDBME)	OS	03
DFHWOS	CSECT	XRF overseer startup module	OS	03
DFHWOSA	CSECT	XRF overseer initialization module	OS	03
DFHWOSB	CSECT	XRF overseer services module	OS	03
DFHWOSM	Macro	XRF overseer interface definition	11	-
DFHWSADS	DSECT	CAVM surveillance status control block	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHWSCDS	DSECT	CAVM time-of-day difference control area	0S	-
DFHWSMDS	DSECT	CAVM state management record	0S	-
DFHWSNDS	DSECT	XRF table of entry points in load module DFHWSMS	0S	-
DFHWSRDS	DSECT	CAVM surveillance communication area	0S	-
DFHWSRTR	CSECT	CAVM state management request router and subtask entry point	0S	03
DFHWSSDS	DSECT	CAVM state management parameter block	0S	-
DFHWSSN1	CSECT	CAVM state management signon initial entry point	0S	03
DFHWSSN2	CSECT	CAVM state management signon request handler	0S	03
DFHWSSN3	CSECT	CAVM state management data set initialization routine	0S	03
DFHWSSOF	CSECT	CAVM state management sign-off request handler	0S	03
DFHWSSR	CSECT	CAVM surveillance status reader	0S	03
DFHWSSW	CSECT	CAVM surveillance status writer	0S	03
DFHWSTDS	DSECT	XRF takeover parameter area	0S	-
DFHWSTI	CSECT	CAVM surveillance tick generator and system status monitor	0S	03
DFHWSTKV	CSECT	CAVM state management takeover request handler	0S	03
DFHWSXDS	DSECT	NOTIFY exit control block	0S	-
DFHWSXPI	CSECT	CAVM state management CAVM process initialization	0S	03
DFHWS2DS	DSECT	Parameter list for DFHWSSN2	0S	-
DFHWS3DS	DSECT	Parameter list for DFHWSSN3	0S	-
DFHWTADS	DSECT	XRF takeover initiation argument block	0S	-
DFHWTI	CSECT	XRF takeover initiation program	0S	03
DFHWTIA	Source	XRF takeover initiation program - RST specific routines	0S	-
DFHWTIC	Source	XRF takeover initiation program - CLT specific routines	0S	-
DFHWTII	Source	XRF takeover initiation program - inquire job status	0S	-
DFHWTIJ	Source	XRF takeover initiation program - job termination/wait	0S	-
DFHWTO	Macro	Write to console operator	11	-
DFHWTRP	CSECT	XRF trace routine	0S	03
DFHXBMDS	Macro	BMS User Exits Parameter List	11	-
DFHXCALL	Macro	EXCI EXEC Interface	11	-
DFHXCDMP	CSECT (OCO)	EXCI dump services	-	03
DFHXCEIP	CSECT (OCO)	EXCI EXEC API handler	-	03
DFHXCGUR	CSECT	EXCI Get Unit of Recovery Tokens	-	03
DFHXCO	Macro	EXCI EXEC options	11	-
DFHXCOPT	DSECT	EXCI options table	19	03
DFHXCP	CSECT	Transaction manager (part)	0S	03
DFHXCPLD	Sample	EXCI CALL parameter list (Assembler)	11	-
DFHXCPLH	Sample	EXCI CALL parameter list (C)	-	08
DFHXCPLL	Sample	EXCI CALL parameter list (PL/I)	-	17
DFHXCPLO	Sample	EXCI CALL parameter list (COBOL)	-	07
DFHXCPRH	DSECT	EXCI program request handler	-	03
DFHXCRCD	Sample	EXCI return codes (Assembler)	11	-
DFHXCRCH	Sample	EXCI return codes (C)	D2	08
DFHXCRCL	Sample	EXCI return codes (PL/I)	-	17
DFHXCRCO	Sample	EXCI return codes (COBOL)	-	07
DFHXCSTB	CSECT	EXCI stub	-	03
DFHXCSVC	CSECT (OCO)	EXCI SVC services	-	03
DFHXCTAB	CSECT (OCO)	EXCI language table	-	03
DFHXCTRA	CSECT	EXCI global trap program	11	03
DFHXCTRD	DSECT	EXCI global trap program parameter list	11	-
DFHXCTRI	CSECT	EXCI trace initialization termination, and recovery	-	03
DFHXCTRP	CSECT	EXCI trace services	-	03
DFHXCURM	CSECT	EXCI user-replaceable module	19	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHXDTDS	Sample	Data Table User Exits Parameter List	11	-
DFHXDXDF	CSECT	DU domain - transaction dump formatter for headers and general information	OS	03
DFHXFDL	Macro	DL/I function shipping	OS	-
DFHXFFC	Macro	FC function shipping	OS	-
DFHXFHED	Macro	Produce transformation program headings	OS	-
DFHXFIC	Macro	IC function shipping	OS	-
DFHXFIOA	DSECT	Transformer I/O area	OS	-
DFHXFJC	Macro	JC function shipping	OS	-
DFHXFMOD	Macro	Produce data transformation programs	OS	-
DFHXFP	CSECT	Online data transformation program	OS	03
DFHXFPC	Macro	DFHXFMOD inner macro	OS	-
DFHXFQ	CSECT	Batch data transformation program	OS	03
DFHXFQU	Macro	TD and TS function shipping	OS	-
DFHXFRM	Macro	Function shipping recovery module	-	03
DFHXFSM	Macro	DFHXFMOD inner macro	OS	-
DFHXFSTG	Macro	XF control block and transformer	11	-
DFHXFX	CSECT	Optimized data transformation program	OS	03
DFHXIS	Sample	XISCONA global user exit program	19	03
DFHXISDS	Sample	XISCONA data set information	19	-
DFHXLT	Macro	Transaction list table	11	-
DFHXLTDS	DSECT	Transaction list table	OS	-
DFHXMAB	CSECT (OCO)	XM domain - abend handler	-	03
DFHXMACT	CSECT		-	03
DFHXMAT	CSECT (OCO)	XM domain - attach	-	03
DFHXMATA	Source	XMAT parameter list	OS	-
DFHXMATM	Source	XMAT request	OS	-
DFHXMATT	CSECT (OCO)	XMAT trace interpretation data	-	03
DFHXMBD	CSECT (OCO)	XM domain - browse	-	03
DFHXMBDA	Source	XMBD parameter list	OS	-
DFHXMBDM	Source	XMBD request	OS	-
DFHXMBDT	CSECT (OCO)	XMBD trace interpretation data	-	03
DFHXMCDS	DSECT	XM domain - TCLASS statistics	11	-
DFHXMCDS	DSECT	XM domain - TCLASS statistics	C2	07
DFHXMCL	CSECT (OCO)	XM domain - transaction class functions	-	03
DFHXMCLA	Source	XMCL parameter list	OS	-
DFHXMCLM	Source	XMCL request	OS	-
DFHXMCLT	CSECT (OCO)	XMCL trace interpretation data	-	03
DFHXMCLX	Macro	XMCL request	11	-
DFHXMCLY	DSECT	XMCL parameter list	11	-
DFHXMDD	CSECT (OCO)	XM domain - delete installed transaction	-	03
DFHXMDDA	Source	XMDD parameter list	OS	-
DFHXMDDM	Source	XMDD request	OS	-
DFHXMDDT	CSECT (OCO)	XMDD trace interpretation data	-	03
DFHXMDM	CSECT (OCO)	XM domain - pre-initialize, initialize, and quiesce domain functions	-	03
DFHXMDNA	Source	XMDN parameter list	OS	-
DFHXMDNT	CSECT	XMDN trace interpretation data	-	03
DFHXMDUF	CSECT (OCO)	Transaction manager SDUMP formatter	-	03
DFHXMER	CSECT (OCO)	XM domain - XMER gate functions	-	03
DFHXMERA	Source	XMER parameter list	OS	-
DFHXMERM	Source	XMER request	OS	-
DFHXMERT	CSECT	XMER trace interpretation data	-	03
DFHXMFD	CSECT (OCO)	XM domain - XMFD gate functions	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHXMFDA	Source	XMFD parameter list	OS	-
DFHXMFDM	Macro	XMFD requests	OS	-
DFHXMFDT	CSECT (OCO)	XMFD trace interpretation data	-	03
DFHXMGDS	DSECT	XM domain - global statistics	11	-
DFHXMGDS	DSECT	XM domain - global statistics	C2	07
DFHXMIQ	CSECT (OCO)	XM domain - XMIQ gate functions	-	03
DFHXMIQA	Source	XMIQ parameter list	OS	-
DFHXMIQI	Source	XMIQ request (inline form of DFHXMIQM)	OS	-
DFHXMIQM	Source	XMIQ requests	OS	-
DFHXMIQT	CSECT (OCO)	XMIQ trace interpretation data	-	03
DFHXMIQX	Macro	XMIQ requests	11	-
DFHXMIQY	DSECT	XMIQ parameter list	11	-
DFHXMLD	CSECT (OCO)	XM domain - XMLD gate functions	-	03
DFHXMLDA	Source	XMLD parameter list	OS	-
DFHXMLDM	Source	XMLD requests	OS	-
DFHXMLDT	CSECT	XMLD trace interpretation data	-	03
DFHXMNTA	DSECT	XMNT parameter list	OS	-
DFHXMNTT	CSECT	XMNT trace interpretation data	-	03
DFHxphPA	DSECT	xphP parameter list	OS	-
DFHxphPT	CSECT	xphP trace interpretation data	-	03
DFHXMQC	CSECT (OCO)	XM domain - tclass functions subroutine	-	03
DFHXMQCA	Source	XMQC parameter list	OS	-
DFHXMQCM	Source	XMQC request	OS	-
DFHXMQCT	CSECT	XMQC trace interpretation data	-	03
DFHXMQD	CSECT (OCO)	XM domain - quiesce and delete transaction definitions functions subroutine	-	03
DFHXMQDT	CSECT (OCO)	XMQD trace interpretation data	-	03
DFHXMRDS	DSECT	XM domain - transaction statistics	11	-
DFHXMRDS	DSECT	XM domain - transaction statistics	C2	07
DFHXMRM	CSECT	XM domain Run Transaction Syncpoint Process.	-	03
DFHXMRM1	CSECT		-	03
DFHXMRP	CSECT (OCO)	XM domain - definition recovery subroutine	-	03
DFHXMRPT	CSECT (OCO)	XMRP trace interpretation data	-	03
DFHXMRSD	DSECT (OCO)	XM domain - communications area for transaction restart (Assembler)	11	-
DFHXMRSH	DSECT (OCO)	XM domain - communications area for transaction restart (C/370)	-	08
DFHXMRSL	DSECT (OCO)	XM domain - communications area for transaction restart (PL/I)	-	17
DFHXMRSO	DSECT (OCO)	XM domain - communications area for transaction restart (COBOL)	-	07
DFHXMRU	CSECT	XMRU CDURUN and Gate Module	-	03
DFHXMRUT	CSECT		-	03
DFHXMSG	CSECT	Default XRF recovery message	OS	03
DFHXMSR	CSECT (OCO)	XM domain - XMSR gate functions	-	03
DFHXMSRA	Source	XMSR parameter list	OS	-
DFHXMSRM	Source	XMSR request	OS	-
DFHXMSRT	CSECT (OCO)	XMSR trace interpretation data	-	03
DFHXMSRX	Macro	XMSR request	11	-
DFHXMSRY	DSECT	XMSR parameter list	11	-
DFHXMST	CSECT (OCO)	XM domain - statistics services	-	03
DFHXMSUA	DSECT	XMSU parameter list	OS	-
DFHXMSUM	Macro	XMSU request	OS	-
DFHXMSUT	CSECT	XMSU trace interpretation data	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHXMTA	CSECT (OCO)	XM domain - task reply gate	-	03
DFHXMTRI	CSECT (OCO)	XM domain - trace initialization, termination, and recovery	-	03
DFHXMTRM	Macro	Obtain 3 character task number from TCA of task issuing trace put	OS	-
DFHXMXD	CSECT (OCO)	XM domain - XMXD gate functions	-	03
DFHXMXDA	Source	XMXD parameter list	OS	-
DFHXMXDD	Source	XMXD transaction definition instance parameter list	OS	-
DFHXMXDI	Source	XMXD request (inline form of DFHXMXDM)	OS	-
DFHXMXDM	Source	XMXD request	OS	-
DFHXMXDT	CSECT (OCO)	XMXD trace interpretation data	-	03
DFHXMXDX	Macro	XMXD request	11	-
DFHXMXDY	DSECT	XMXD parameter list	11	-
DFHXMXE	CSECT (OCO)	XM domain - XMXE gate functions	-	03
DFHXMXEA	Source	XMXE parameter list	OS	-
DFHXMXEM	Source	XMXE request	OS	-
DFHXMXET	CSECT (OCO)	XMXE trace interpretation data	-	03
DFHXMXM	CSECT	Run Transaction XM Attach Client	-	03
DFHXMXND	CSECT (OCO)	XM domain - transaction storage	OS	-
DFHXOPU@	CSECT		-	03
DFHXOPUS	Sample	Sample IIOP URM (C Version)	-	19
DFHXQBF	CSECT	XQ queue server buffer pool routines	-	03
DFHXQCF	CSECT	XQ queue server coupling facility I/O	-	03
DFHXQCN	CSECT	XQ queue server connect/disconnect	-	03
DFHXQDF	CSECT	XQ TS queue pool server definitions	-	03
DFHXQEN	CSECT	XQ ENF event interface	-	03
DFHXQIF	CSECT	XQ queue server interface module	-	03
DFHXQIQ	CSECT	XQ queue server inquire module	-	03
DFHXQMN	CSECT	XQ queue server mainline	-	03
DFHXQMS	CSECT	XQ queue pool server messages	-	03
DFHXQOP	CSECT	XQ queue server command processing	-	03
DFHXQPR	CSECT	XQ queue server parameter processing	-	03
DFHXQRL	CSECT	XQ queue server reload routine	-	03
DFHXQRQ	CSECT	XQ queue server request routine	-	03
DFHXQRS	CSECT	XQ ARM Restart Support	-	03
DFHXQST	CSECT	XQ queue server statistics	-	03
DFHXQS1D	CSECT	XQ list structure statistics record	11	-
DFHXQS2D	CSECT	XQ queue buffer statistics record	11	-
DFHXQS3D	CSECT	XQ main storage statistics record	11	-
DFHXQUL	CSECT	XQ queue server unload routine	-	03
DFHXR	Macro	XRF code generation macro	11	-
DFHXRA	CSECT	XRF request processing program	OS	03
DFHXRB	CSECT	XRF NOTIFY exit program	OS	03
DFHXRC	CSECT	XRF inquire status exit program	OS	03
DFHXRCP	CSECT	XRF console communication program	OS	03
DFHXRDUF	CSECT (OCO)	XRF SDUMP formatter	-	03
DFHXRE	CSECT	XRF startup program	OS	03
DFHXRF	CSECT	XRF CAVM sign-off interface	OS	03
DFHXRHDS	DSECT	XRF health data definition	11	-
DFHXROCL	Other	Used by DFHCRST cataloged procedure	11	-
DFHXRSP	CSECT	XRF surveillance program	OS	03
DFHXRXDF	CSECT	DU domain - transaction dump formatter for XRF related areas	OS	03
DFHXSAD	CSECT (OCO)	XS domain - XSAD gate functions	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHXSADA	Source	XSAD parameter list	OS	-
DFHXSADM	Source	XSAD request	OS	-
DFHXSADT	CSECT (OCO)	XSAD trace interpretation data	-	03
DFHXSDM	CSECT (OCO)	XS domain - initialize, quiesce, terminate domain functions	-	03
DFHXSDUF	CSECT (OCO)	XS domain - SDUMP formatter	-	03
DFHXSEAI	CSECT	Early verification stub program	-	03
DFHXSEJ	CSECT	Security Interfaces for EJB	-	03
DFHXSEJT	CSECT		-	03
DFHXSEV	CSECT (OCO)	XS domain - early verification support	-	03
DFHXSFL	CSECT (OCO)	XS domain - XSFL gate functions	-	03
DFHXSFLA	Source	XSFL parameter list	OS	-
DFHXSFLM	Source	XSFL request	OS	-
DFHXSFLT	CSECT (OCO)	XSFL trace interpretation data	-	03
DFHXSIDT	CSECT (OCO)	XS domain - trace interpretation data	-	03
DFHXSIS	CSECT (OCO)	XS domain - XSIS gate functions	-	03
DFHXSISA	Source	XSIS parameter list	OS	-
DFHXSISM	Source	XSIS request	OS	-
DFHXSIST	CSECT (OCO)	XSIS trace interpretation data	-	03
DFHXSLU	CSECT (OCO)	XS domain - XSLU gate functions	-	03
DFHXSLUA	Source	XSLU parameter list	OS	-
DFHXSLUM	Source	XSLU request	OS	-
DFHXSLUT	CSECT (OCO)	XSLU trace interpretation data	-	03
DFHXSPUB	DSECT (OCO)	XS domain - public storage fields	OS	-
DFHXSPW	CSECT (OCO)	XS domain - XSPW gate functions	-	03
DFHXSPWA	Source	XSPW parameter list	OS	-
DFHXSPWM	Source	XSPW request	OS	-
DFHXSPWT	CSECT (OCO)	XSPW trace interpretation data	-	03
DFHXSRC	CSECT (OCO)	XS domain - XSRC gate functions	-	03
DFHXSRCA	Source	XSRC parameter list	OS	-
DFHXSRCI	Source	XSRC request (inline form of DFHXSRCM)	OS	-
DFHXSRCM	Macro	XSRC requests	OS	-
DFHXSRCT	CSECT (OCO)	XSRC trace interpretation data	-	03
DFHXSSA	CSECT (OCO)	XS domain - supervisor request router	-	03
DFHXSSAT	CSECT (OCO)	XSSA trace interpretation data	-	03
DFHXSSB	CSECT (OCO)	XS domain - supervisor extraction services	-	03
DFHXSSBT	CSECT (OCO)	XSSB trace interpretation data	-	03
DFHXSSC	CSECT (OCO)	XS domain - resource checking functions	-	03
DFHXSSCT	CSECT (OCO)	XSSC trace interpretation data	-	03
DFHXSSD	CSECT (OCO)	XS domain - create passticket function	-	03
DFHXSSDT	CSECT (OCO)	XSSD trace interpretation data	-	03
DFHXSSE	CSECT	Security Supervisor Phase E Cert.Mgement	-	03
DFHXSSET	CSECT		-	03
DFHXSSI	CSECT (OCO)	XS domain - storage initialization	-	03
DFHXSSIT	CSECT (OCO)	XSSI trace interpretation data	-	03
DFHXSTRI	CSECT (OCO)	XS domain - trace initialization, termination, and recovery	-	03
DFHXSUXP	Macro	Installation data for ESM exits	11	-
DFHXSWM	CSECT	XRF message manager for security manager	OS	03
DFHXSWMA	CSECT	XSWM parameter list	OS	-
DFHXSWMM	Macro	XSWM request	OS	-
DFHXSXM	CSECT (OCO)	XS domain - XM domain interface	-	03
DFHXSXMA	DSECT	XSXM parameter list	OS	-
DFHXSXMI	Macro	XSXM requests (inline form)	OS	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHXSXMM	Macro	XSXM requests	OS	-
DFHXSXMT	CSECT (OCO)	XSXM trace interpretation data	-	03
DFHXT	Macro	DFHXTP internal table generator	OS	-
DFHXTAB	Macro	BMS internal macro	11	-
DFHXTCI	CSECT	XRF terminal switching	OS	03
DFHXTENF	Sample	XICTENF/XALTENF global user exit program	19	03
DFHXTEP	CSECT	User-replaceable terminal error program	19	03
DFHXTEPT	CSECT	User-replaceable terminal error tables	19	03
DFHXTP	CSECT	Terminal sharing transformation program	OS	03
DFHXTPD	DSECT	XTP internal control blocks	OS	-
DFHXTSTG	Macro	XTP parameter list	OS	-
DFHXTT	Source	XTP data transformation argument descriptions (used by DFHXT macro)	OS	-
DFHXTTT	Macro	DFHXTT inner macro	OS	-
DFHXZIDS	DSECT	XZIQUE exit data set information	11	-
DFHYBTPL	Other	Cataloged procedure to translate, compile, and link-edit Language Environment PL/I application programs that use EXEC DLI and will run in a batch or CICS shared database region	18	-
DFHYBTVL	Other	Cataloged procedure to translate, compile, and link-edit Language Environment COBOL application programs that use EXEC DLI and will run in a batch or CICS shared database region	18	-
DFHYITDL	Other	Cataloged procedure to translate, compile, and link-edit Language Environment C application programs	18	-
DFHYITEL	Other	Cataloged procedure to translate, compile, and link-edit C++ application programs using the Language Environment compiler	18	-
DFHYITPL	Other	Cataloged procedure to translate, compile, and link-edit Language Environment PL/I application programs	18	-
DFHYITVL	Other	Cataloged procedure to translate, compile, and link-edit Language Environment COBOL application programs	18	-
DFHYXTDL	Other	Cataloged procedure to translate, compile, and link-edit Language Environment C application programs that are to use the external CICS interface	18	-
DFHYXTEL	Other	Cataloged procedure to translate (EXCI), compile, and link-edit C++ application programs using the Language Environment compiler	18	-
DFHYXTPL	Other	Cataloged procedure to translate, compile, and link-edit Language Environment PL/I application programs that are to use the external CICS interface	18	-
DFHYXTVL	Other	Cataloged procedure to translate, compile, and link-edit Language Environment COBOL application programs that are to use the external CICS interface	18	-
DFHZABD	CSECT	No VTAM support abend handler	OS	03
DFHZACT	CSECT	Activate scan	OS	03
DFHZAIT	CSECT	Attach initialization table	OS	-
DFHZAND	CSECT	Abend control block	OS	03
DFHZAPB	Sample	3770 application program	19	-
DFHZARER	CSECT	LU6.2 protocol error and exception handler	OS	03
DFHZARL	CSECT	LU6.2 application request logic	OS	03
DFHZARM	CSECT	LU6.2 migration logic	OS	03
DFHZARQ	CSECT	Application request handler	OS	03
DFHZARR	CSECT	LU6.2 application receive request logic	OS	03
DFHZARRA	CSECT	LU6.2 application receive buffer support	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHZARRC	CSECT	LU6.2 classify what next to receive	OS	03
DFHZARRF	CSECT	LU6.2 receive FMH7 and ER1	OS	03
DFHZASX	CSECT	DFASY exit	OS	03
DFHZATA	CSECT	Autoinstall program	OS	03
DFHZATA2	CSECT	ZCINST Autoinstall Program - Console	-	03
DFHZATD	CSECT	Autoinstall delete program	OS	03
DFHZATDX	CSECT	User-replaceable autoinstall exit	19	03
DFHZATDY	CSECT	User-replaceable autoinstall exit with APPC	19	03
DFHZATI	CSECT	Automatic task initiation	OS	03
DFHZATMD	CSECT	Automatic terminal remote definition program	-	03
DFHZATMF	CSECT	Mass flag program for time-out delete	-	03
DFHZATR	CSECT	Autoinstall restart program	OS	03
DFHZATS	CSECT	Remote autoinstall/delete program	OS	03
DFHZATT	CSECT	Task attach	OS	03
DFHZBAN	CSECT	Terminal control bind analysis	OS	03
DFHZBKT	CSECT	LU6.2 bracket state machine	OS	03
DFHZBLX	CSECT	VTAM SCIP exit LU6.2 bind handling	OS	03
DFHZBSM	Macro	LU6.2 bracket state macro	OS	-
DFHZCA	CSECT	VTAM working set module	OS	03
DFHZCB	CSECT	VTAM working set module	OS	03
DFHZCC	CSECT	VTAM working set module	OS	03
DFHZCGRP	CSECT (OCO)	Attach CGRP task (for DFHZGRP)	-	03
DFHZCHM	Macro	LU6.2 chain state macro	OS	-
DFHZCHS	CSECT	LU6.2 chain state machine	OS	03
DFHZCLS	CSECT	CLSDST	OS	03
DFHZCLX	CSECT	CLSDST exit	OS	03
DFHZCNA	CSECT	System console activity control	OS	03
DFHZCNM	Macro	LU6.2 contention state macro	OS	-
DFHZCNR	CSECT	System console application request	OS	03
DFHZCNT	CSECT	LU6.2 contention state machine	OS	03
DFHZCNVM	Macro	MRO application state setting	OS	-
DFHZCN1	CSECT	CICS Client CCIN Transaction	-	03
DFHZCN2	CSECT	CICS Client CCIN ZC domain subroutine	-	03
DFHZCN2T	DSECT	ZCN2 translate tables	-	03
DFHZCTR1	CSECT	ZC CICS Client trace interpretation	-	03
DFHZCOVR	CSECT	Terminal control open VTAM retry	-	03
DFHZCP	CSECT	Terminal management program	OS	03
DFHZCPBK	Macro	Bracket control	OS	-
DFHZCPLR	CSECT	PL/AS call for TCPLR	OS	03
DFHZCQ	Macro	Terminal control install interface	11	-
DFHZCQCH	CSECT	Catalog a TCT element	OS	03
DFHZCQDL	CSECT	Dynamic delete TCT element	OS	03
DFHZCQIN	CSECT	Initialize DFHZCQ	OS	03
DFHZCQIQ	CSECT	Inquire about a TCTTE	OS	03
DFHZCQIS	CSECT	Install a TCTTE	OS	03
DFHZCQRS	CSECT	Restore a terminal control resource	OS	03
DFHZCQRT	CSECT	ZC resource types table	OS	03
DFHZCQ00	CSECT	Dynamic add/replace TCT elements	OS	03
DFHZCRM	Macro	LU6.2 RPL_B state macro	OS	-
DFHZCRQ	CSECT	CTYPE command request	OS	03
DFHZCRT	CSECT	LU6.2 RPL_B state machine	OS	03
DFHZCSTP	CSECT	Attach CSTP (TCP task)	OS	03
DFHZCTDX	Sample	Autoinstall user exit - COBOL	-	19
DFHZCTRI	CSECT	Persistent sessions trace interpreter	-	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHZCT1	CSECT	CICS Client CTIN transaction	-	03
DFHZCUT	CSECT	Persistent verification signed-on-from list management program	OS	03
DFHZCUTA	DSECT	ZCUT parameter list	OS	-
DFHZCUTM	Macro	ZCUT request	OS	-
DFHZCUTT	CSECT	ZCUT trace interpretation data	OS	03
DFHZCW	CSECT	VTAM nonworking set module	OS	03
DFHZCX	CSECT	LOCATE, ISC/IRC request	OS	03
DFHZCXR	CSECT	Transaction routing module address list	OS	03
DFHZCY	CSECT	VTAM nonworking set module	OS	03
DFHZCZ	CSECT	VTAM nonworking set module	OS	03
DFHZDET	CSECT	Task detach	OS	03
DFHZDSP	CSECT	Dispatcher	OS	03
DFHZDST	CSECT	SNA-ASCII translator	OS	03
DFHZDTDX	Sample	Autoinstall user exit - C/370	D3	-
DFHZEMW	CSECT	Error message writer	OS	03
DFHZEPD	DSECT	TCP/ZCP module entry address list	11	-
DFHZEQU	Symbolic	ZCP equates	11	-
DFHZERH	CSECT	LU6.2 error program	OS	03
DFHZERRM	Macro	ZCP error-handling macro	OS	-
DFHZETR	Macro	ZC VTAM exit GTF trace macro	OS	-
DFHZEV1	CSECT	LU6.2 security encryption program part 1	OS	03
DFHZEV2	CSECT	LU6.2 security encryption program part 2	OS	03
DFHZFRE	CSECT	FREEMAIN request	OS	03
DFHZGAI	CSECT (OCO)	APPC autoinstall - create APPC clones	-	03
DFHZGAIA	Source	ZGAI parameter list	OS	-
DFHZGAIM	Source	ZGAI request	OS	-
DFHZGAIT	CSECT	ZGAI trace interpretation data	OS	03
DFHZGBM	CSECT (OCO)	APPC manipulate bitmap	-	03
DFHZGBMA	Source	ZGBM parameter list	OS	-
DFHZGBMM	Source	ZGBM request	OS	-
DFHZGBMT	CSECT (OCO)	ZGBM trace interpretation data	-	03
DFHZGCA	CSECT (OCO)	LU6.2 CNOS actioning	-	03
DFHZGCAA	Source	ZGCA parameter list	OS	-
DFHZGCAM	Source	ZGCA request	OS	-
DFHZGCAT	CSECT (OCO)	ZGCA trace interpretation data	-	03
DFHZGCC	CSECT (OCO)	Catalog CNOS services	-	03
DFHZGCCA	Source	ZGCC parameter list	OS	-
DFHZGCCM	Source	ZGCC request	OS	-
DFHZGCCT	CSECT (OCO)	ZGCC trace interpretation data	-	03
DFHZGCH	CSECT	ZC VTAM change macro domain subroutine	-	03
DFHZGCHA	CSECT	ZGCH parameter list	OS	-
DFHZGCHM	Macro	ZGCH request	OS	-
DFHZGCHT	DSECT	ZGCH translate tables	OS	03
DFHZGCN	CSECT (OCO)	LU6.2 CNOS negotiation	-	03
DFHZGCNA	Source	ZGCN parameter list	OS	-
DFHZGCNM	Source	ZGCN request	OS	-
DFHZGCNT	CSECT (OCO)	ZGCN trace interpretation data	-	03
DFHZGDA	CSECT (OCO)	VTAM persistent sessions deallocate abend functions	-	03
DFHZGDAA	Source	ZGDA parameter list	OS	-
DFHZGDAM	Macro	ZGDA requests	11	-
DFHZGDAT	CSECT (OCO)	ZGDA trace interpretation data	-	03
DFHZGDCD	CSECT	Terminal control subroutine constants	OS	-
DFHZGET	CSECT	GETMAIN request	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHZGIN	CSECT	ZC VTAM INQUIRE domain subroutine	-	03
DFHZGINA	CSECT	ZGIN parameter list	OS	-
DFHZGINM	Macro	ZGIN request	OS	-
DFHZGINT	DSECT	ZGIN translate tables	-	03
DFHZGPC	CSECT (OCO)	LU6.2 recover CNOS values for modegroups	-	03
DFHZGPCA	Source	ZGPC parameter list	OS	-
DFHZGPCM	Source	ZGPC request	OS	-
DFHZGPCT	CSECT (OCO)	ZGPC trace interpretation data	-	03
DFHZGPR	CSECT (OCO)	VTAM persistent sessions resource handler	-	03
DFHZGPRA	Source	ZGPR parameter list	OS	-
DFHZGPRI	Source	ZGPR request (inline form of DFHZGPRM)	OS	-
DFHZGPRM	Source	ZGPR request	OS	-
DFHZGPRT	CSECT (OCO)	ZGPR trace interpretation data	-	03
DFHZGRP	CSECT (OCO)	VTAM persistent sessions initialization	-	03
DFHZGRPA	Source	ZGRP parameter list	OS	-
DFHZGRPD	Source	ZGRP control blocks	OS	-
DFHZGRPM	Source	ZGRP request	OS	-
DFHZGRPT	CSECT (OCO)	ZGRP trace interpretation data	-	03
DFHZGSL	CSECT (OCO)	VTAM persistent sessions set logon	-	03
DFHZGSLA	Source	ZGSL parameter list	OS	-
DFHZGSLM	Source	ZGSL request	OS	-
DFHZGSLT	CSECT (OCO)	ZGSL trace interpretation data	-	03
DFHZGTA	CSECT	ZC TMP table alter gate	-	03
DFHZGTAA	CSECT	ZGTA parameter list	OS	-
DFHZGTAM	Macro	ZGTA request	OS	-
DFHZGTAT	DSECT	ZGTA translate tables	-	03
DFHZGTI	CSECT	ZC TMP table inquire gate	-	03
DFHZGTIA	CSECT	ZGTI parameter list	OS	-
DFHZGTIC	CSECT	ZGTI create copybook	OS	-
DFHZGTIM	Macro	ZGTI request	OS	-
DFHZGTIT	DSECT	ZGTI translate tables	-	03
DFHZGTRA	DSECT	ZGTR interface parameter area	OS	-
DFHZGTRM	Macro	DFHZGTR interface macro	OS	-
DFHZGTRT	DSECT	ZGTR translate tables	-	03
DFHZGUB	CSECT (OCO)	VTAM persistent sessions terminate	-	03
DFHZGUBA	Source	ZGUB parameter list	OS	-
DFHZGUBM	Source	ZGUB request	OS	-
DFHZGUBT	CSECT (OCO)	ZGUB trace interpretation data	-	03
DFHZGURD	CSECT (OCO)	VTAM persistent sessions URD table	OS	-
DFHZGXA	CSECT	LU6.2 extended attach security	-	03
DFHZGXAA	DSECT	ZGXA parameter list	OS	-
DFHZGXAM	Macro	ZGXA requests	OS	-
DFHZGXAT	CSECT	ZGXA trace interpretation data	OS	03
DFHZHPCH	Macro	Generate authorized path CHECK or CHECK macro	OS	-
DFHZHPDS	DSECT	ZCP call plist for initialization of SRB facility (HPO)	OS	-
DFHZHPRV	Macro	Generate authorized path RECEIVE or RECEIVE macro	OS	-
DFHZHPRX	CSECT	Authorized path SRB mode VTAM EXECRPL	OS	03
DFHZHPSD	Macro	Generate authorized path SEND or SEND macro	OS	-
DFHZHPSR	CSECT	Authorized path SRB requests	OS	03
DFHZINT	Source	Terminal control initialization	OS	-
DFHZISP	CSECT	Allocate/free/point	OS	03
DFHZIS1	CSECT	Prepare/SPR/commit/abend	OS	03
DFHZIS2	CSECT	IRC internal requests	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHZLEX	CSECT	LERAD exit	OS	03
DFHZLGX	CSECT	Logon exit	OS	03
DFHZLOC	CSECT	Locate	OS	03
DFHZLRP	CSECT	Logical record presentation	OS	03
DFHZLS1	CSECT	LU6.2 CNOS request transaction program	-	03
DFHZLS1M	Macro	LU6.2 CNOS request	OS	-
DFHZLTX	CSECT	LOSTERM exit	OS	03
DFHZMJM	Macro	NACP sense code table generation macro	OS	-
DFHZNAC	CSECT	Node abnormal condition program (NACP)	OS	03
DFHZNCA	CSECT	NACP message table generator	OS	-
DFHZNCE	CSECT	NACP interface to NEP	OS	-
DFHZNCM	Macro	NACP message table generation macro	OS	-
DFHZNCS	CSECT	Sense code analysis	OS	-
DFHZNCV	CSECT	VTAM return code analysis	OS	-
DFHZNEPI	Macro	NEP interface generator	11	-
DFHZNEPX	Source	Translated command-level default NEP	19	-
DFHZNEP0	CSECT	User-replaceable node error program	19	03
DFHZNSET	Other	SMP/E zone setter (used by cataloged procedures)	11	-
DFHZNSP	CSECT	VTAM services procedure error exit	OS	03
DFHZOPA	CSECT	Dynamic VTAM open	OS	03
DFHZOPN	CSECT	OPNDST	OS	03
DFHZOPX	CSECT	OPNDST exit	OS	03
DFHZPTDX	Sample	Autoinstall user exit - PL/I	-	19
DFHZQUE	CSECT	Attach chain and queue subroutine	OS	03
DFHZRAC	CSECT	Receive-any completion	OS	03
DFHZRAQ	CSECT	Read ahead queuing	OS	03
DFHZRAR	CSECT	Read ahead retrieval	OS	03
DFHZRAS	CSECT	Receive-any slowdown processing	OS	03
DFHZRBDS	DSECT	LU6.2 application receive set buffer hdr	OS	-
DFHZRLP	CSECT	LU6.2 post-VTAM receive logic	OS	03
DFHZRLX	CSECT	LU6.2 receive exit program	OS	03
DFHZRPL	Source	TC build receive-any RPLs	OS	-
DFHZRQM	Macro	Add element to RPL completion queue	11	-
DFHZRRX	CSECT	Release request exit	OS	03
DFHZRSP	CSECT	Resync send program	OS	03
DFHZRST	CSECT	RESETSR	OS	03
DFHZRSY1	CSECT	VTAM LU6.1 resynchronization	-	03
DFHZRSY2	CSECT	VTAM LU6.1 resynchronization	-	03
DFHZRSY3	CSECT	VTAM LU6.1 resynchronization	-	03
DFHZRSY4	CSECT	VTAM LU6.1 resynchronization	-	03
DFHZRSY5	CSECT	VTAM LU6.1 resynchronization	-	03
DFHZRSY6	CSECT	VTAM LU6.1 resynchronization	-	03
DFHZRTRI	CSECT	VTAM LU6.1 resynchronization trace interpretation	-	03
DFHZRVL	CSECT	LU6.2 pre-VTAM receive logic	OS	03
DFHZRVS	CSECT	Receive specific	OS	03
DFHZRVX	CSECT	Receive specific exit	OS	03
DFHZSAX	CSECT	Send DFASY exit	OS	03
DFHZSCX	CSECT	Session control input exit	OS	03
DFHZSDA	CSECT	Send asynchronous command	OS	03
DFHZSDL	CSECT	LU6.2 send logic	OS	03
DFHZSDR	CSECT	Send response	OS	03
DFHZSDS	CSECT	Send DFSYN	OS	03
DFHZSDX	CSECT	Send DFSYN data exit	OS	03
DFHZSES	CSECT	SESSIONC	OS	03

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFHZSEX	CSECT	SESSIONC exit	OS	03
DFHZSHU	CSECT	Checks shutdown status for VTAM terminals	OS	03
DFHZSIM	CSECT	SIMLOGON	OS	03
DFHZSIX	CSECT	SIMLOGON exit	OS	03
DFHZSKR	CSECT	Command response	OS	03
DFHZSLDS	Symbolic	Send list data structure	11	-
DFHZSLS	CSECT	Set logon start	OS	03
DFHZSLX	CSECT	LU6.2 send exit program	OS	03
DFHZSSX	CSECT	Send DFSYN exit	OS	03
DFHZSTAM	Macro	DFHZSTAP interface	OS	-
DFHZSTAP	CSECT	Conversation state determination	OS	03
DFHZSTU	CSECT	Terminal control status change	OS	03
DFHZSUP	CSECT	Startup task	OS	03
DFHZSYN	CSECT	VTAM recovery module	OS	03
DFHZSYX	CSECT	SYNAD exit	OS	03
DFHZS1DS	DSECT	ZC SUBPOOL_TOKENs table	OS	-
DFHZTAX	CSECT	Turnaround exit	OS	03
DFHZTPX	CSECT	TPEND exit	OS	03
DFHZTR	Macro	ZCP trace macro	OS	-
DFHZTRA	CSECT	VTAM trace module	OS	03
DFHZTSP	CSECT	Terminal sharing program	OS	03
DFHZUCT	CSECT	Uppercase translate	OS	03
DFHZUIX	CSECT	User input exit	OS	03
DFHZUSR	CSECT	LU6.2 conversation state machine	OS	03
DFHZUSRM	Macro	LU6.2 conversation state macro	OS	-
DFHZXCU	CSECT	VTAM XRF catch-up transaction	OS	03
DFHZXDUF	CSECT (OCO)	XRF ZCP queue SDUMP formatter	-	03
DFHZXPS	CSECT	VTAM persistent sessions APPC recovery	-	03
DFHZXQO	CSECT	XRF ZCP tracking queue organizer	OS	03
DFHZXQOS	Symbolic	DFHZXQO internal control blocks	OS	-
DFHZXRC	CSECT	XRF and Persistent sessions state data analysis	OS	03
DFHZXRE0	CSECT	VTAM reconnect transaction	OS	03
DFHZXRL	CSECT	Transaction routing - LU6.2 command processor, AOR	OS	03
DFHZXRPL	Macro	Clear RPL	OS	-
DFHZXRT	CSECT	Transaction routing - LU6.2 command processor, TOR	OS	03
DFHZXS	Macro	Interface to DFHZXST	OS	-
DFHZXST	CSECT	XRF ZCP session-state tracking	OS	03
DFHZXSTS	CSECT	SETLOGON routine	OS	03
DFH0AZBC	Sample	FEPI sample: CICS back-end application	19	-
DFH0AZBI	Sample	FEPI sample: IMS back-end application	19	-
DFH0AZPA	Sample	FEPI sample: SLU P pseudo-conversational program (Assembler)	19	-
DFH0AZPS	Sample	FEPI sample: SLU P one-out one-in program (Assembler)	19	-
DFH0AZQS	Sample	FEPI sample: STSN processing	19	-
DFH0AZTD	Sample	FEPI sample: 3270 data stream pass through	19	-
DFH0AZXS	Sample	FEPI sample: setup program (Assembler)	19	-
DFH0BAT1	Sample	Batch enabling sample BAT1 - disable transactions coordinator	C3	-
DFH0BAT2	Sample	Batch enabling sample BAT2 - inquire retained locks coordinator	C3	-
DFH0BAT3	Sample	Batch enabling sample BAT3 - force retained locks coordinator	C3	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFH0BAT4	Sample	Batch enabling sample BAT1 - disable transactions program	C3	-
DFH0BAT5	Sample	Batch enabling sample BAT2 - inquire retained locks program	C3	-
DFH0BAT6	Sample	Batch enabling sample BAT3 - force indoubt UOWs program	C3	-
DFH0BAT7	Sample	Batch enabling sample BAT2 - retry backout failures program	C3	-
DFH0BAT8	Sample	Batch enabling sample BAT3 - forcibly release locks program	C3	-
DFH0BCA	Sample	CUA communication area layout - COBOL	C3	-
DFH0BCR	Sample	CUA customer record layout - COBOL	C3	-
DFH0BC11	Sample	Batch enabling sample BAT1 - disable transactions TS queue	C3	-
DFH0BC12	Sample	Batch enabling sample BAT1 - disable transactions commarea	C3	-
DFH0BC21	Sample	Batch enabling sample BAT2 - inquire retained locks TS queue	C3	-
DFH0BC22	Sample	Batch enabling sample BAT2 - inquire retained locks commarea	C3	-
DFH0BC23	Sample	Batch enabling sample BAT2 - inquire retained locks map texts	C3	-
DFH0BC31	Sample	Batch enabling sample BAT3 - force retained locks TS queue	C3	-
DFH0BC32	Sample	Batch enabling sample BAT3 - force retained locks commarea	C3	-
DFH0BFKT	Sample	CUA variable function key layout - COBOL	C3	-
DFH0BFPD	Sample	CUA redefinition of file pull-down - COBOL	C3	-
DFH0BHP	Sample	CUA redefinition of help pop-up - COBOL	C3	-
DFH0BHPD	Sample	CUA redefinition of help pull-down - COBOL	C3	-
DFH0BHR	Sample	CUA help text TS queue layout - COBOL	C3	-
DFH0BHT	Sample	CUA help file key table - COBOL	C3	-
DFH0BLST	Sample	CUA redefinition of list base panel - COBOL	C3	-
DFH0BMSG	Sample	CUA application message table - COBOL	C3	-
DFH0BM1	Sample	Batch enabling sample BAT1 - disable transactions BMS mapset	19	-
DFH0BM1O	Sample	Batch enabling sample BAT1 - disable transactions BMS mapset	C3	-
DFH0BM2	Sample	Batch enabling sample BAT2 - inquire retained locks BMS mapset	19	-
DFH0BM2O	Sample	Batch enabling sample BAT2 - inquire retained locks BMS mapset	C3	-
DFH0BM3	Sample	Batch enabling sample BAT3 - force retained locks BMS mapset	19	-
DFH0BM3O	Sample	Batch enabling sample BAT3 - force retained locks BMS mapset	C3	-
DFH0BRT	Sample	CUA program routing control table - COBOL	C3	-
DFH0BTSQ	Sample	CUA TS queue details layout - COBOL	C3	-
DFH0BZCA	Sample	FEPI sample: system definition and customization (Assembler)	19	-
DFH0BZCC	Sample	FEPI sample: system definition and customization (C/370)	D3	-
DFH0BZCO	Sample	FEPI sample: system definition and customization (COBOL)	C3	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFH0BZCP	Sample	FEPI sample: system definition and customization (PL/I)	P3	-
DFH0BZMA	Sample	FEPI sample: messages and text (Assembler)	19	-
DFH0BZMC	Sample	FEPI sample: messages and text (C/370)	D3	-
DFH0BZMO	Sample	FEPI sample: messages and text (COBOL)	C3	-
DFH0BZMP	Sample	FEPI sample: messages and text (PL/I)	P3	-
DFH0BZ1O	Sample	FEPI sample: front-end terminal map (COBOL)	C3	-
DFH0BZ2O	Sample	FEPI sample: front-end terminal map (COBOL)	C3	-
DFH0BZ3A	Sample	FEPI sample: front-end terminal map (Assembler)	19	-
DFH0BZ4O	Sample	FEPI sample: front-end terminal map (COBOL)	C3	-
DFH0BZ5O	Sample	FEPI sample: front-end terminal map (COBOL)	C3	-
DFH0BZ6C	Sample	FEPI sample: front-end terminal map (C/370)	D3	-
DFH0BZ7P	Sample	FEPI sample: front-end terminal map (PL/I)	P3	-
DFH0BZ8A	Sample	FEPI sample: front-end terminal map	19	-
DFH0BZ9A	Sample	FEPI sample: front-end terminal map	19	-
DFH0CALL	Sample	Inquiry/update - COBOL	C3	-
DFH0CBAC	Sample	Sample CICS BTS 3270 Transaction Client	C3	-
DFH0CBAE	Sample	Sample Bridge Exit Routine	C3	-
DFH0CBAI	Sample	Sample input routine for BTS 3270 txn	C3	-
DFH0CBAO	Sample	Sample output routine for BTS 3270 txn	C3	-
DFH0CBDC	Sample	CSD backup program - COBOL	C3	-
DFH0CBRD	Sample	Sample bridge exit common area	C3	-
DFH0CBRE	Sample	Sample bridge exit	C3	-
DFH0CBRF	Sample	Sample bridge formatter	C3	-
DFH0CBRU	Sample	Sample bridge exit user area	C3	-
DFH0CBRW	Sample	Browse - COBOL	C3	-
DFH0CCOM	Sample	Order entry queue print - COBOL	C3	-
DFH0CESD	Sample	Shutdown assist program - COBOL	C3	-
DFH0CFIL	Sample	Customer file (FILEA) record layout - COBOL	C3	-
DFH0CLOG	Sample	Audit trail (log) record layout - COBOL	C3	-
DFH0CL86	Sample	Order entry queue record layout - COBOL	C3	-
DFH0CMA	Sample	Operator instructions map set - COBOL	19	-
DFH0CMB	Sample	Customer details map set - COBOL	19	-
DFH0CMC	Sample	File browse map set - COBOL	19	-
DFH0CMD	Sample	Low balance inquiry map set - COBOL	19	-
DFH0CMK	Sample	Order entry map set - COBOL	19	-
DFH0CML	Sample	Order report map set - COBOL	19	-
DFH0CMNU	Sample	Operator instructions - COBOL	C3	-
DFH0CONT	Sample	sample program for CICS BTS	C3	-
DFH0CMP	Sample	Keystroke overlap/look-aside query - map set - COBOL	19	-
DFH0CPKO	Sample	Keystroke overlap - COBOL	-	19
DFH0CPLA	Sample	Look-aside query - COBOL	-	19
DFH0CREN	Sample	Order entry - COBOL	-	19
DFH0CREP	Sample	Low balance inquiry - COBOL	-	19
DFH0CRFC	Sample	CSD cross-reference program - COBOL	-	19
DFH0CXCC	Sample	Keystroke overlap - COBOL	-	19
DFH0CZTK	Sample	FEPI sample: keystroke CONVERSE program (C/370)	D3	19
DFH0CZXS	Sample	FEPI sample: setup program (C/370)	D3	19
DFH0DCUS	Sample	CUA customer details file contents	05	-
DFH0DEL1	Sample	Sample program for CICS BTS	-	19
DFH0DHLP	Sample	CUA help file contents	04	-
DFH0DHTX	Sample	Sample EXITPGM Template	-	19
DFH0DLCC	Sample	CICS-DL/I program (CALL) - COBOL	-	19
DFH0DLCE	Sample	CICS-DL/I program (EXEC) - COBOL	-	19

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFH0FORC	Sample	DB2 formatting program - COBOL	-	19
DFH0GMAP	Sample	Sample goodnight program map set	-	19
DFH0GNIT	Sample	Sample goodnight transaction	-	19
DFH0INV1	Sample	Sample program for CICS BTS	-	19
DFH0IZRI	Sample	FEPI sample: RDO data for back-end IMS	19	-
DFH0IZRQ	Sample	FEPI sample: RDM data for front-end CICS	19	-
DFH0JCUS	Other	JCL to create CUA customer details file	02	-
DFH0JHLP	Other	JCL to create CUA help file	02	-
DFH0MAB	Sample	CUA abend handling - map set - COBOL	19	-
DFH0MABT	Sample	CUA about the sample application pop-up - map set - COBOL	19	-
DFH0MBRW	Sample	CUA browse customer details, base panel - map set - COBOL	19	-
DFH0MDEL	Sample	CUA delete a customer record, base panel - map set - COBOL	19	-
DFH0MFPD	Sample	CUA file pull-down - map set - COBOL	19	-
DFH0MHLP	Sample	CUA help stub full-screen pop-up - map set - COBOL	19	-
DFH0MHP	Sample	CUA contextual help pop-up - map set - COBOL	19	-
DFH0MHPD	Sample	CUA help pull-down - map set - COBOL	19	-
DFH0MLST	Sample	CUA list processing, base panel - map set - COBOL	19	-
DFH0MNEW	Sample	CUA new customer record, base panel - map set - COBOL	19	-
DFH0MOPN	Sample	CUA file open pop-up - map set - COBOL	19	-
DFH0MPRT	Sample	CUA print pop-up - map set - COBOL	19	-
DFH0MSAS	Sample	CUA save changed customer record pop-up - map set - COBOL	19	-
DFH0MT1	Sample	CUA primary panel for sample application - map set - COBOL	19	-
DFH0MUPD	Sample	CUA update customer details, base panel - map set - COBOL	19	-
DFH0MZ1	Sample	FEPI sample: keystroke CONVERSE map (COBOL)	19	-
DFH0MZ2	Sample	FEPI sample: send/start and receive map (COBOL)	19	-
DFH0MZ3	Sample	FEPI sample: map for back-end CICS application (Assembler)	19	-
DFH0MZ4	Sample	FEPI sample: SLU P one-out one-in map (COBOL)	19	-
DFH0MZ5	Sample	FEPI sample: SLU P pseudo-conversational map (COBOL)	19	-
DFH0MZ6	Sample	FEPI sample: keystroke CONVERSE map (C/370)	19	-
DFH0MZ7	Sample	FEPI sample: keystroke CONVERSE map (PL/I)	19	-
DFH0MZ8	Sample	FEPI sample: SLU P one-out one-in map (Assembler)	19	-
DFH0MZ9	Sample	FEPI sample: SLU P pseudo-conversational map (Assembler)	19	-
DFH0PAYC	Sample	Sample program for CICS BTS	19	-
DFH0PAYM	Sample	Sample program for CICS BTS	19	-
DFH0PAY0	Sample	Sample program for CICS BTS	19	-
DFH0PAY1	Sample	Sample program for CICS BTS	19	-
DFH0PS	Sample	Keystroke overlap/look-aside query - partition set - COBOL	19	-
DFH0PZTK	Sample	FEPI sample: keystroke CONVERSE program (PL/I)	-	19
DFH0RED1	Sample	Sample program for CICS BTS	19	-
DFH0REM1	Sample	Sample program for CICS BTS	19	-
DFH0SALC	Sample	Sample program for CICS BTS	19	-
DFH0SALM	Sample	Sample program for CICS BTS	19	-
DFH0SAL0	Sample	Sample program for CICS BTS	19	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFH0SAL1	Sample	Sample program for CICS BTS	19	-
DFH0SAL2	Sample	Sample program for CICS BTS	19	-
DFH0SET	Sample	Menu map for sample application	19	-
DFH0SINX	Sample	Rebuild primer index from master file	-	19
DFH0SIXR	Sample	Name index record for sample application	-	19
DFH0SREC	Sample	Account file record for sample application	-	19
DFH0STAT	Sample	Collect and print statistics - COBOL	C3	03
DFH0STCM	Sample	Statistics sample (DFH0STAT) Commarea	-	19
DFH0STLK	Sample		C3	03
DFH0STM	Sample	Collect and print stats map set - COBOL	19	03
DFH0STMD	Sample		19	-
DFH0STMU	Sample		19	-
DFH0STOC	Sample	Sample program for CICS BTS	19	-
DFH0STPR	Sample	Sample program for CICS BTS	19	03
DFH0STS	Sample	Statistics sample mapset - report selection	19	03
DFH0STSD	Sample		19	-
DFH0STSU	Sample		19	-
DFH0STSY	Sample		19	03
DFH0STTP	Sample		19	03
DFH0S00	Sample	Online account menu sample program	C3	-
DFH0S01	Sample	File inquire for sample application	C3	-
DFH0S02	Sample	File update for sample application	C3	-
DFH0S03	Sample	Print customer record for sample application	C3	-
DFH0S04	Sample	Error routine for sample application	C3	-
DFH0VAB	Sample	CUA abend handler - COBOL	C3	-
DFH0VABT	Sample	CUA about pop-up handler - COBOL	C3	-
DFH0VBRW	Sample	CUA browse customer details processing - COBOL	C3	-
DFH0VDEL	Sample	CUA delete customer details processing - COBOL	C3	-
DFH0VDQ	Sample	CUA temporary-storage cleanup - COBOL	C3	-
DFH0VHLP	Sample	CUA help pop-up handler - COBOL	C3	-
DFH0VHP	Sample	CUA contextual help pop-up handler - COBOL	C3	-
DFH0VLIO	Sample	CUA help file handler - COBOL	C3	-
DFH0VLST	Sample	CUA list panel handler - COBOL	C3	-
DFH0VNEW	Sample	CUA new customer panel processing - COBOL	C3	-
DFH0VOL	Sample	CUA overlay handler - COBOL	C3	-
DFH0VOPN	Sample	CUA file open pop-up handler - COBOL	C3	-
DFH0VPRT	Sample	CUA print pop-up handler - COBOL	C3	-
DFH0VRIO	Sample	CUA customer detail file handler - COBOL	C3	-
DFH0VSAS	Sample	CUA save customer details pop-up handler - COBOL	C3	-
DFH0VTBL	Sample	CUA table router - COBOL	C3	-
DFH0VT1	Sample	CUA primary panel processing - COBOL	C3	-
DFH0VUPD	Sample	CUA update customer record processing - COBOL	C3	-
DFH0VZPA	Sample	FEPI sample: SLU P pseudo-conversational program (COBOL)	C3	-
DFH0VZPS	Sample	FEPI sample: SLU P one-out one-in program (COBOL)	C3	-
DFH0VZQS	Sample	FEPI sample: STSN handler (COBOL)	C3	-
DFH0VZTD	Sample	FEPI sample: 3270 data stream pass through (COBOL)	C3	-
DFH0VZTK	Sample	FEPI sample: keystroke CONVERSE program (COBOL)	C3	-
DFH0VZTR	Sample	FEPI sample: screen image RECEIVE/ EXTRACT FIELD (COBOL)	C3	-
DFH0VZTS	Sample	FEPI sample: screen image SEND/START (COBOL)	C3	-
DFH0VZUC	Sample	FEPI sample: begin session handler (COBOL)	C3	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFH0VZUU	Sample	FEPI sample: end session handler (COBOL)	C3	-
DFH0VZUX	Sample	FEPI sample: monitor unsolicited data handler (COBOL)	C3	-
DFH0VZXS	Sample	FEPI sample: setup program (COBOL)	C3	-
DFH0WBCA	Sample	Sample Client Authentication Program	-	19
DFH2980	Symbolic	Special characters for 2980	C2	07
DFH3QSS	Sample		OS	03
DFH62XM	Sample	62 XM transaction attach	-	03
DFH99BC	Sample	Dynamic allocation - convert to binary target	19	03
DFH99BLD	Other	Dyn alloc - JCL to build sample program	02	-
DFH99CC	Sample	Dyn alloc - character and numeric string conversion	19	03
DFH99DY	Sample	Dyn alloc - issue SVC and analyze	19	03
DFH99FP	Sample	Dyn alloc - process function keyword	19	03
DFH99GI	Sample	Dyn alloc - format display and get input	19	03
DFH99KC	Sample	Dyn alloc - keyword value conversion	19	03
DFH99KH	Sample	Dyn alloc - list keywords for help	19	03
DFH99KO	Sample	Dyn alloc - process operator keywords	19	03
DFH99KR	Sample	Dyn alloc - convert returned value to keyword	19	03
DFH99LK	Sample	Dyn alloc - search key set for given token	19	03
DFH99M	Sample	Dyn alloc - macro	11	-
DFH99MAC	Sample	Dyn alloc - macro	19	-
DFH99ML	Sample	Dyn alloc - build message text from token list	19	03
DFH99MM	Sample	Dyn alloc - main control program	19	03
DFH99MP	Sample	Dyn alloc - message filing routine	19	03
DFH99MT	Sample	Dyn alloc - match abbreviation with keyword	19	03
DFH99RP	Sample	Dyn alloc - process returned values	19	03
DFH99SVC	Sample	Dyn alloc - SVC services	19	-
DFH99T	Sample	Dyn alloc - table of keywords	19	03
DFH99TK	Sample	Dyn alloc - tokenize input command	19	03
DFH99TX	Sample	Dyn alloc - text display routine	19	03
DFH99VH	Sample	Dyn alloc - list description for help	19	03
DFH$AALL	Sample	Inquiry/update	19	03
DFH$ABRW	Sample	Browse	19	03
DFH$ACOM	Sample	Order entry queue print	19	03
DFH$ADSP	Sample	XRF overseer - display status	19	03
DFH$AFIL	Sample	Customer file (FILEA) record layout	19	-
DFH$AGA	Sample	Generated version of DFH$AMA	19	03
DFH$AGB	Sample	Generated version of DFH$AMB	19	03
DFH$AGC	Sample	Generated version of DFH$AMC	19	03
DFH$AGCB	Sample	XRF overseer - set up RPL	19	03
DFH$AGD	Sample	Generated version of DFH$AMD	19	03
DFH$AGK	Sample	Generated version of DFH$AMK	19	03
DFH$AGL	Sample	Generated version of DFH$AML	19	03
DFH$ALOG	Sample	Audit trail (log) record layout	19	-
DFH$AL86	Sample	Order entry queue record layout	19	-
DFH$AMA	Sample	Operator instructions map set	19	-
DFH$AMAU	Sample	Operator instructions map set	19	-
DFH$AMB	Sample	Customer details map set	19	-
DFH$AMBU	Sample	Customer details map set	19	-
DFH$AMC	Sample	File browse map set	19	-
DFH$AMCU	Sample	File browse map set	19	-
DFH$AMD	Sample	Low balance inquiry map set	19	-
DFH$AMDU	Sample	Web Interface BMS screen emulation	19	-
DFH$AMK	Sample	Order entry map set	19	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFH$AMKU	Sample	Order entry map set	19	-
DFH$AML	Sample	Order report map set	19	-
DFH$AMNU	Sample	Operator instructions	19	03
DFH$AREN	Sample	Order entry	19	03
DFH$AREP	Sample	Low balance inquiry	19	03
DFH$ARES	Sample	XRF overseer - restart failed region	19	03
DFH$ATXC	Sample	EXCI batch client program (Assembler)	19	03
DFH$AXCC	Sample	EXCI batch client program (Assembler)	19	03
DFH$AXCS	Sample	EXCI batch server program (Assembler)	19	03
DFH$AXRO	Sample	XRF overseer program	19	03
DFH$AXVS	Sample	EXCI sample server	19	03
DFH$BMXT	Sample	Sample BMS global user exit	19	-
DFH$BTCH	Sample	Batch test data for DFHIVPBT	19	-
DFH$CAT1	Sample	CLIST to create RACF profiles for CICS category 1 transactions	19	-
DFH$CAT2	Sample	CLIST to create RACF profiles for CICS category 2 transactions	19	-
DFH$CESD	Sample	Shutdown assist program	P3	-
DFH$CRFA	Sample	CSD cross-reference program	19	03
DFH$CRFP	Sample	CSD cross-reference program - PL/I	P3	-
DFH$CSDU	Sample	RDO offline utilities	19	-
DFH$CUS1	Sample	CSDUP invocation from TSO environment	19	03
DFH$DALL	Sample	Inquiry/update - C/370	D3	-
DFH$DBAN	Sample	Batch test data for DFHIVPDB (Assembler)	19	-
DFH$DBCB	Sample	Batch test data for DFHIVPDB (Cobol)	19	-
DFH$DBPL	Sample	Batch test data for DFHIVPDB (PL/I)	19	-
DFH$DBRW	Sample	Browse - C/370	D3	-
DFH$DB2T	Sample	DB2 table definitions for DFH$FORx	19	-
DFH$DCOM	Sample	Order entry queue print - C/370	D3	-
DFH$DCTD	Sample	DCT SDSCI entries	19	-
DFH$DCTR	Sample	DCT entries for basic facilities	19	-
DFH$DCTS	Sample	DCT entries for sample applications	19	-
DFH$DFIL	Sample	Customer file (FILEA) record layout -C/370	D3	-
DFH$DLAC	Sample	CICS-DL/I program using CALL interface	19	03
DFH$DLAE	Sample	CICS-DL/I program using EXEC DLI	19	03
DFH$DLPC	Sample	CICS-DL/I program (CALL) - PL/I	P3	-
DFH$DLPE	Sample	CICS-DL/I program (EXEC) - PL/I	P3	-
DFH$DL86	Sample	Order entry queue record layout - C/370	D3	-
DFH$DMA	Sample	Operator instructions map set - C/370	19	-
DFH$DMB	Sample	Customer details map set - C/370	19	-
DFH$DMC	Sample	File browse map set - C/370	19	-
DFH$DMD	Sample	Low balance inquiry map set - C/370	19	-
DFH$DMK	Sample	Order entry map set - C/370	19	-
DFH$DML	Sample	Order report map set - C/370	19	-
DFH$DMNU	Sample	Operator instructions - C/370	D3	-
DFH$DREN	Sample	Order entry - C/370	D3	-
DFH$DREP	Sample	Low balance inquiry - C/370	D3	-
DFH$DTLC	Sample	Shared Data Tables XDTLC exit program	19	-
DFH$DTAD	Sample	Shared data tables XDTAD exit program	19	-
DFH$DTRD	Sample	Shared data tables XDTRD exit program	19	-
DFH$DXCC	Sample	Batch Client Program (C/370)	D3	-
DFH$DXVC	Sample	EXCI client program - Java environment	19	03
DFH$FAIN	Sample	Data for batch load of FILEA	19	-
DFH$FCBF	Sample	Sample XFCBFAIL exit program	19	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFH$FCBV	Sample	Sample XFCBOVER exit program	19	-
DFH$FCLD	Sample	Sample XFCLDEL exit program	19	-
DFH$FORA	Sample	DB2 formatting program	19	03
DFH$FORP	Sample	DB2 formatting program - PL/I	P3	-
DFH$GMAP	Sample	Sample goodnight transaction BMS map	19	-
DFH$ICCN	Sample	Call to CPSM to issue cancel command	19	-
DFH$ICIC	Sample	CICS-CICS or CICS-IMS conversation	19	03
DFH$IFBL	Sample	Remote file browse - local processing	19	03
DFH$IFBR	Sample	Remote file browse - remote processing	19	03
DFH$IGB	Sample	Generated version of DFH$IMB	19	03
DFH$IGC	Sample	Generated version of DFH$IMC	19	03
DFH$IGS	Sample	Generated version of DFH$IMS	19	03
DFH$IGX	Sample	Generated version of DFH$IMX	19	03
DFH$IG1	Sample	Generated version of DFH$IM1	19	03
DFH$IG2	Sample	Generated version of DFH$IM2	19	03
DFH$IIAT	Sample	IIOP banking sample app.to C Account	19	-
DFH$IIBI	Sample	IIOP banking sample app.to C Init.	19	-
DFH$IIBQ	Sample	IIOP banking sample app.to C Query	19	-
DFH$IICC	Sample	IIOP banking sample app.to C Credit Check	19	-
DFH$IICH	Sample	IIOP banking sample app.to C Cr.Chk commarea	19	-
DFH$IIMA	Sample	IIOP banking sample app.to C BMS Map	19	-
DFH$IIQR	Sample	IIOP banking sample app.to C Comm_struct	19	-
DFH$IMB	Sample	Remote file browse - map set	19	-
DFH$IMC	Sample	CICS-CICS or CICS-IMS conversation - map set	19	-
DFH$IMS	Sample	CICS-IMS conversation/demand paged output - map set	19	-
DFH$IMSN	Sample	CICS-IMS conversation	19	03
DFH$IMSO	Sample	CICS-IMS demand paged output	19	03
DFH$IMX	Sample	Local to remote temporary-storage queue transfer - map set	19	-
DFH$IM1	Sample	TS record retrieval - map set 1	19	-
DFH$IM2	Sample	TS record retrieval - map set 2	19	-
DFH$IQRD	Sample	TS record retrieval - local display	19	03
DFH$IQRL	Sample	TS record retrieval - local request	19	03
DFH$IQRR	Sample	TS record retrieval - remote request	19	03
DFH$IQXL	Sample	Local to remote temporary-storage queue transfer - local processing	19	03
DFH$IQXR	Sample	Local to remote temporary-storage queue transfer - remote processing	19	03
DFH$JSAM	Sample	Java Sample Linker in C	19	-
DFH$LCCA	Sample	Java Sample COMMAREA checker in C	19	-
DFH$LDSP	Sample	Create FILEA data file	19	03
DFH$LGLS	Sample	Sample GLUE program for XLGSTRM	19	-
DFH$MCTD	Sample	MCT entry for DBCTL	19	-
DFH$MOLS	Sample	Offline processor of monitoring data	19	03
DFH$OFAR	Sample		19	-
DFH$PALL	Sample	Inquiry/update - PL/I	P3	-
DFH$PBRW	Sample	Browse - PL/I	P3	-
DFH$PCEX	Sample	XPCFTCH global user exit program	19	03
DFH$PCGA	Sample	Global work area for DFH$PCEX	19	-
DFH$PCOM	Sample	Order entry queue print - PL/I	P3	-
DFH$PCPI	Sample	Enabling program for DFH$PCEX and DFH$ZCAT	19	03
DFH$PCPL	Sample	DFH$PCEX global user exit invocation	19	03
DFH$PCTA	Sample	XPCTA user exit program	19	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFH$PDUM	Sample	Dummy main program for PL/I programs using CSD offline extract function	P3	-
DFH$PFIL	Sample	Customer file (FILEA) record layout - PL/I	P3	-
DFH$PLOG	Sample	Audit trail (log) record layout - PL/I	P3	-
DFH$PL86	Sample	Order entry queue record layout - PL/I	P3	-
DFH$PMA	Sample	Operator instructions map set - PL/I	19	-
DFH$PMB	Sample	Customer details map set - PL/I	19	-
DFH$PMC	Sample	File browse map set - PL/I	19	-
DFH$PMD	Sample	Low balance inquiry map set - PL/I	19	-
DFH$PMK	Sample	Order entry map set - PL/I	19	-
DFH$PML	Sample	Order report map set - PL/I	19	-
DFH$PMNU	Sample	Operator instructions - PL/I	P3	-
DFH$PMP	Sample	Keystroke overlap/look-aside query - map set - PL/I	19	-
DFH$PPKO	Sample	Keystroke overlap - PL/I	P3	-
DFH$PPLA	Sample	Look-aside query - PL/I	P3	-
DFH$PREN	Sample	Order entry - PL/I	P3	-
DFH$PREP	Sample	Low balance inquiry PL/I	P3	-
DFH$PS	Sample	Keystroke overlap/look-aside query - partition set - PL/I	19	-
DFH$PXCC	Sample	Batch client program (PL/I)	P3	-
DFH$RACF	Sample	RACF class descriptor table	19	-
DFH$RING	Sample	Build KEYRING profiles in RACF	19	-
DFH$SIPA	Other	System initialization parameters for use with AOR and default SIT	19	-
DFH$SIPD	Other	System initialization parameters for use with DOR and default SIT	19	-
DFH$SIPT	Other	System initialization parameters for use with TOR and default SIT	19	-
DFH$SIP1	Other	System initialization parameters for use by DFHIVPOL (online IVP)	19	-
DFH$SIP2	Other	System initialization parameters for use by DFHIVPBT (batch IVP)	19	-
DFH$SIP5	Other	System initialization parameters for use by DFHIVPDB (DBCTL IVP)	19	-
DFH$SNPW	Sample	Password expiration mgement for Windows/NT	19	-
DFH$SNP2	Sample	Password expiration mgement for OS/2 Warp	19	-
DFH$SQLT	Sample	Input for DB2 table load utility	19	-
DFH$STAS	Sample	DFH0STAT storage statistics subroutine	19	03
DFH$STCN	Sample	DFH0STAT time calculations subroutine	19	03
DFH$STED	Sample	Stagger end-of-day time for statistics	19	03
DFH$STER	Sample	PLT program to print recovery statistics on CICS emergency restart	19	03
DFH$STTB	Sample	Statistics sample user exit ID table	19	03
DFH$SXP1	Sample	Suppress message by number (user exit)	19	03
DFH$SXP2	Sample	Suppress message by destination route code	19	03
DFH$SXP3	Sample	Suppress message by transient data queue	19	03
DFH$SXP4	Sample	Reroute console message to transient data queue	19	03
DFH$SXP5	Sample	Reroute message from one transient data queue to another	19	03
DFH$SXP6	Sample	Reroute message from transient data queue to list of consoles	19	03
DFH$TCTS	Sample	TCT entries for sequential (CRLP) terminals	19	-
DFH$TDWT	Sample	Transient data write to terminal	19	03
DFH$ULPA	Other	Placeholder for DFH$ULPA	19	-

Table 123. CICS modules directory (continued)

Name	Type	Description	Library	
DFH$UMOD	Other	SMP/E USERMOD to move LPA-eligible CICS modules into LPA library	19	-
DFH$WBAU	Sample	Web module	19	03
DFH$WBSA	Sample	Web module	19	03
DFH$WBSB	Sample	Web module	19	03
DFH$WBSC	Sample	Web module	19	03
DFH$WBSN	Sample	Web module	19	03
DFH$WBSR	Sample	Web module	19	03
DFH$WBST	Sample	Web module	19	03
DFH$WB1A	Sample	Web module	19	03
DFH$WB1C	Sample	Web module	D3	-
DFH$XDRQ	Sample		19	03
DFH$XNQE	Sample	Exec ENQ/DEQ Sample XNQEREQ Exit	19	03
DFH$XRDS	Sample	XRF overseer control blocks	19	-
DFH$XTSE	Sample	XTSEREQ global user exit program	19	03
DFH$XZIQ	Sample	Sample XZIQUE global user exit program	19	03
DFH$ZCAT	Sample	Sample XZCATT global user exit program	19	03
DFH$ZCGA	Sample	Global work area for DFH$ZCAT	19	-
DFJ$UMOD	Sample	Place holder for DFJ$UMOD	19	-
DLIUIB	DSECT	DL/I user interface block	C2	-
DLIUIB	DSECT	DL/I user interface block	P2	-
DLIUIB	DSECT	DL/I user interface block	D3	-
DLIUIB	Macro	DL/I user interface block	12	-
DFH99SVC	CSECT	Dyn alloc - SVC services	-	03
DSNCPRMA	Macro	CICS-DB2 connect dynamic plan selection parmlist (Assembler)	12	-
DSNCPRMC	Macro	CICS-DB2 connect dynamic plan selection parmlist (COBOL)	C2	-
DSNCPRMP	Macro	CICS-DB2 connect dynamic plan selection parmlist (PL/I)	P2	-
DSNCRCT	Macro	CICS-DB2 connect RCT macro	12	-
DSNCUEXT	CSECT	CICS-DB2 connect dynamic plan selection	19	03
MEUKEYS	CSECT	MEU key definitions	17	-
MEULANG	CSECT	MEU language table	11	-
MEU00	CSECT	MEU MEU00x message set	13	-
MEU01	CSECT	MEU MEU01x message set	13	-
MEU02	CSECT	MEU MEU02x message set	13	-
MEU03	CSECT	MEU MEU03x message set	13	-
MEU04	CSECT	MEU MEU04x message set	13	-
MEU05	CSECT	MEU MEU05x message set	13	-
SRRC	Symbolic	SAA resource recovery pseudonyms for C	D3	-
SRRCOBOL	Symbolic	SAA resource recovery pseudonyms for COBOL	C2	-
SRRHASM	Symbolic	SAA resource recovery pseudonyms for assembler-language	12	-
SRRPLI	Symbolic	SAA pseudonym file for PL/I	P2	-

Chapter 116. CICS executable modules

The following list shows, for each module:

1. The name of the module
2. Its entry points
3. Callers of the module
4. A brief description of the module
5. Where the module returns to. This information is omitted where the module returns to its caller (the normal situation).

In general, this list is restricted to non-OCO modules. In the few cases where OCO modules are included, no design details are given.

DFHACP

Entry points: DFHACPNA

Called by: DFHAPRM, DFHAPXME

Description: The abnormal condition program writes a message to the terminal and to the CSMT destination if a transaction abends or cannot be started. Subject to tests on the type of terminal, DFHACP invokes DFHMGP to output the message. It calls DFHPEP and, depending on the result, may disable the transaction. For each error, there is an entry in a table which contains the number of the message to be written to the principal facility (terminal) and the number of the message to be written to CSMT. If, in either case, there is no message, zero is entered.

The main subroutines of DFHACP are:

```
ABCSMTWT - Write to CSMT
ACPCALMG - Use DFHMGP to output a message
ACPCLPEP - Invoke DFHPEP
ACPFENTY - Identify message for terminal
TERMERR  - Terminal error.
```

DFHAICBP

Entry points: DFHAICB

Called by: User application program

Description: The application interface control block program acts both as a control block and, for compatibility with early releases of CICS/VS, as executable code. DFHAICBP provides addressability between application programs and CICS entry points, namely those of the EXEC interface and the common programming interface. DFHAICBP is link-edited with the EXEC interface programs (DFHEIP and DFHEIPA), and the common programming interface program (DFHCPI) to form the application interface program (DFHAIP) load module.

DFHALP

Entry points: DFHALPNA

Called by: DFHCRQ, DFHCRS, DFHICP, DFHTPQ, DFHTPR, DFHTPS, DFHZATI, DFHZISP, DFHZNAC, DFHZTSP

Description: The terminal allocation program contains the logic to allocate TCTTE resources to requesting transactions. The request operates in a multiple exchange between the requesting transaction and terminal control. DFHALP passes a SCHEDULE request to terminal control as an ATI terminal control, then responds with an AVAIL command. The requests are represented by AIDs (AID chain manipulations being performed by calls to DFHALP). For LU6.2, DFHALP issues a terminal control allocate mode name macro.

DFHAMP

Entry points: DFHAMPNA

Called by: DFHEIP, DFHSII1

Description: The allocation management program is invoked by the CEDA transaction. It analyzes commands and calls the definition file management program, DFHDMP, to process changes to records in the CSD. For the INSTALL command, DFHAMP also calls program manager, transaction manager, and DFHSPP. DFHPUP is called to convert data between address list format and the CSD record format.

DFHAPJC

Entry points: DFHAPJCN

Called by: User

Description: The AP domain journal control gate service module handles WRITE_JOURNAL_DATA calls made by the user exit's XPI. It gets a TCA if the task doesn't currently have one, and also a JCA. If the task already has a JCA, this is stacked. It then copies the parameter list passed in the domain call, to the JCA, and then issues one of four journal writes, depending on the request. Finally the return code from the JC write is copied into the domain parameter list, and the JCA and TCA are released if they were obtained by DFHAPJC.

DFHAPSIP

Entry points: DFHSIPNA

Called by: DFHAPDM

Description: The main AP domain initialization program provides DFHWTO support and common subroutines used by DFHSIA1 through DFHSIJ1. In sequence, DFHAPSIP performs the following functions:

- Defines the AP domain subpools
- Acquires the SIT address
- Passes control to the DFHSIA1, DFHSIB1, and so on.

The main subroutines of DFHAPSIP are:

```
CHKRLVLR - Check release level
OVERLSUP - Overlay supervisor
SIGETCOR - Storage allocation
SILOADR  - Program loader
SIPCONS  - Console WRITE.
```

DFHAPST

Entry points: DFHAPST

Called by: DFHEIP, DFHSTST

Description: The supervisory statistics program within the AP domain accepts a request for and then supervises the copying/resetting of statistics counters in the AP domain by calling the appropriate DFHSTxx modules to access the counters.

This module is called when:

- Statistics domain is collecting INTERVAL statistics and calls this module to pass it copies of and to reset all statistics in AP domain. This module then sequentially calls all of the DFHSTxx modules to do the copying and resetting.
- A CEMT PERFORM STATISTICS command results in a call to the statistics domain which then makes an appropriate call to this module to pass it copies of the requested statistics. This module then calls the DFHSTxx modules required to do the copying.
- An EXEC CICS COLLECT STATISTICS command results in a call to this module which then calls the DFHSTxx module required to pass copies of the statistics back to the application program.

Thus, this module is called only by the statistics domain or by DFHEIP.

This module provides two functions:

COLLECT_STATISTICS
: collects statistics for all resources in the AP domain and calls the statistics domain to write them out to the SMF data set.

COLLECT_RESOURCE_STATS
: collects statistics for the named resource type (optionally qualified by the resource identifier) and either copies them to a buffer available through the API, or causes them to be written to the SMF data set.

DFHAPTD

Entry points: DFHAPTD

Called by: DFHETD, DFHTDA, DFHTDB, ME domain

Description: DFHAPTD handles DFHTDTDM macro requests; as such, it provides the transient data gate into the AP domain. DFHTDTDM macro requests are routed from DFHAPTD to DFHTDP using the corresponding DFHTD CTYPE requests.

DFHAPTI

Entry points: DFHAPTI

Called by: the timer domain to handle NOTIFY calls for the application domain.

Description: The DFHAPTO module looks at the token passed by the timer domain and resumes either the DFHAPTI or DFHAPTIX module, as appropriate.

DFHAPTIM

Entry points: DFHAPTIM

Called by: runs as a system task attached by the DFHSII1 module.

Description: The DFHAPTIM module is part of the interval control mechanism. When it first gets control, it suspends itself to wait for an interval control ICE to expire. Interval control uses the timer domain to handle time intervals. When the timer domain detects the expiry of an interval control related interval, it calls the DFHAPTI module, which in turn resumes the DFHAPTIM module. The DFHAPTIM module then makes an "expiry analysis" call to the DFHICP module, which processes any expired ICEs. On return, the DFHAPTIM module suspends itself again to wait for the next ICE to expire.

DFHAPTIX

Entry points: DFHAPTIX

Called by: runs as a system task attached by the DFHSII1 module.

Description: The DFHAPTIX module is part of the interval control mechanism. When it first gets control, it tells the timer domain that it wants to be told every time it is midnight. It then suspends itself to wait for the next midnight. When that occurs, the timer domain calls the DFHAPTI module, which resumes the DFHAPTIX module, which in turn calls the DFHICP module to do midnight processing.

DFHASV

Entry points: DFHASVNA

Called by: DFHCSVC

Description: DFHASV is one of the modules that run under the CICS type 3 SVC. On entry to DFHASV, register 0 contains one of the following request codes:

```
  0 - Paging request
  8 - SRB termination
  9 - HPO initialization
 24 - Monitoring services
 64 - Authorize general purpose subtask TCB
 80 - Issue SDUMP
136 - Bind AP domain.
```

DFHBSIB3

Entry points: DFHBSIB3

Called by: DFHTBSxx

Description: DFHBSIB3 adds BMS 3270 support to a TCT table entry.

DFHBSIZ1

Entry points: DFHBSIZ1

Called by: DFHTBSxx

Description: DFHBSIZ1 adds SCS support to a TCT table entry.

DFHBSIZ3

Entry points: DFHBSIZ3

Called by: DFHTBSxx

Description: DFHBSIZ3 adds DFHZCP 3270 support to a TCT table entry.

DFHBSMIR

Entry points: DFHBSMIR

Called by: DFHTBSxx

Description: DFHBSMIR builds a TCT table entry for a session.

DFHBSMPP

Entry points: DFHBSMPP

Called by: DFHTBSxx

Description: DFHBSMPP builds a TCT table entry for a pipeline pool entry.

DFHBSM61

Entry points: DFHBSM61

Called by: DFHTBSxx

Description: DFHBSM61 builds sessions for an LU6.2 mode group.

DFHBSM62

Entry points: DFHBSM62

Called by: DFHTBSxx

Description: DFHBSM62 builds the mode entry for an LU6.2 mode group.

DFHBSS

Entry points: DFHBSS

Called by: DFHTBSxx

Description: DFHBSS adds a new connection (system entry) to a CICS system.

DFHBSSA

Entry points: DFHBSSA

Called by: DFHTBSxx

Description: DFHBSSA initializes DFHKCP support in a new TCT system entry.

DFHBSSF

Entry points: DFHBSSF

Called by: DFHTBSxx

Description: DFHBSSF initializes the statistics counters in a new TCT system entry.

DFHBSSS

Entry points: DFHBSSS

Called by: DFHTBSxx

Description: DFHBSSS builds security support for a new TCT system entry.

DFHBSSZ

Entry points: DFHBSSZ

Called by: DFHTBSxx

Description: DFHBSSZ builds VTAM interface support for a new TCT system entry.

DFHBSSZB

Entry points: DFHBSSZB

Called by: DFHTBSxx

Description: DFHBSSZB adds a new batch interregion connection to a CICS system.

DFHBSSZG

Entry points: DFHBSSZG

Called by: DFHTBSxx

Description: DFHBSSZG adds a new advanced program-to-program communication (APPC) single-session connection to a CICS system.

DFHBSSZI

Entry points: DFHBSSZI

Called by: DFHTBSxx

Description: DFHBSSZI adds an indirect terminal control system table entry to a CICS system.

DFHBSSZL

Entry points: DFHBSSZL

Called by: DFHTBSxx

Description: DFHBSSZL adds a local terminal control system table entry to a CICS system.

DFHBSSZM

Entry points: DFHBSSZM

Called by: DFHTBSxx

Description: DFHBSSZM introduces a new connection (system) to ZCP.

DFHBSSZP

Entry points: DFHBSSZP

Called by: DFHTBSxx

Description: DFHBSSZP builds an advanced program-to-program communication (APPC) parallel-session connection to a CICS system.

DFHBSSZR

Entry points: DFHBSSZR

Called by: DFHTBSxx

Description: DFHBSSZR builds an MRO session entry.

DFHBSSZS

Entry points: DFHBSSZS

Called by: DFHTBSxx

Description: DFHBSSZS builds an advanced program-to-program communication (APPC) session entry.

DFHBSSZ6

Entry points: DFHBSSZ6

Called by: DFHTBSxx

Description: DFHBSSZ6 builds an LU6.1 connection entry.

DFHBST

Entry points: DFHBST

Called by: DFHTBSxx

Description: DFHBST performs TCTTE initialization common to terminals, pipeline pool entries, and sessions for IRC and ISC.

DFHBSTB

Entry points: DFHBSTB

Called by: DFHTBSxx

Description: DFHBSTB adds support for BMS to a new TCT terminal or session entry.

DFHBSTBL

Entry points: DFHBSTBL

Called by: DFHTBSxx

Description: DFHBSTBL adds support for logical device components (LDCs).

DFHBSTB3

Entry points: DFHBSTB3

Called by: DFHTBSxx

Description: DFHBSTB3 adds partition support to a new TCT terminal or session entry.

DFHBSTC

Entry points: DFHBSTC

Called by: DFHTBSxx

Description: DFHBSTC performs those operations that are executed after the installation of a terminal.

DFHBSTD

Entry points: DFHBSTD

Called by: DFHTBSxx

Description: DFHBSTD adds data interchange program (DFHDIP) support for a new TCT table entry.

DFHBSTE

Entry points: DFHBSTE

Called by: DFHTBSxx

Description: DFHBSTE adds EXEC diagnostic facility (EDF) support for a new TCT table entry.

DFHBSTH

Entry points: DFHBSTH

Called by: DFHTBSxx

Description: DFHBSTH initializes EXEC interface fields for a new TCT table entry.

DFHBSTI

Entry points: DFHBSTI

Called by: DFHTBSxx

Description: DFHBSTI adds interval control program (DFHICP) support for a new TCT table entry.

DFHBSTM

Entry points: DFHBSTM

Called by: DFHTBSxx

Description: DFHBSTM adds message generation program (DFHMGP) support for a new TCT table entry.

DFHBSTO

Entry points: DFHBSTO

Called by: DFHTBSxx

Description: DFHBSTO is the spooler builder.

DFHBSTP3

Entry points: DFHBSTP3

Called by: DFHTBSxx

Description: DFHBST adds 3270-copy support for a new TCT table entry.

DFHBSTS

Entry points: DFHBSTS

Called by: DFHTBSxx

Description: DFHBSTS adds signon program (DFHSNP) support for a new TCT table entry.

DFHBSTT

Entry points: DFHBSTT

Called by: DFHTBSxx

Description: DFHBSTT adds task control program (DFHKCP) support for a new TCT table entry.

DFHBSTZ

Entry points: DFHBSTZ

Called by: DFHTBSxx

Description: DFHBSTZ builds a session or terminal resource.

DFHBSTZA

Entry points: DFHBSTZA

Called by: DFHTBSxx

Description: DFHBSTZA adds DFHZCP activity scan support to a new TCT terminal or session entry.

DFHBSTZB

Entry points: DFHBSTZB

Called by: DFHTBSxx

Description: DFHBSTZB appends or deletes a BIND image for a TCT terminal or session entry.

DFHBSTZC

Entry points: DFHBSTZC

Called by: DFHTBSxx

Description: DFHBSTZC adds a single-session LU6.2 system as an advanced program-to-program communication (APPC) terminal.

DFHBSTZE

Entry points: DFHBSTZE

Called by: DFHTBSxx

Description: DFHBSTZE sets error message writer fields for a new TCT table entry.

DFHBSTZH

Entry points: DFHBSTZH

Called by: DFHTBSxx

Description: DFHBSTZH adds an interregion (IRC) batch session to a CICS system.

DFHBSTZL

Entry points: DFHBSTZL

Called by: DFHTBSxx

Description: DFHBSTZL adds logical device code support to a new TCT terminal or session entry.

DFHBSTZO

Entry points: DFHBSTZO

Called by: DFHTBSxx

Description: DFHBSTZO adds an MVS console to a CICS system.

DFHBSTZP

Entry points: DFHBSTZP

Called by: DFHTBSxx

Description: DFHBSTZP adds a pipeline pool entry to a CICS system.

DFHBSTZR

Entry points: DFHBSTZR

Called by: DFHTBSxx

Description: DFHBSTZR adds an interregion (IRC) session to a CICS system.

DFHBSTZS

Entry points: DFHBSTZS

Called by: DFHTBSxx

Description: DFHBSTZS adds an advanced program-to-program communication (APPC) session to the terminal control program.

DFHBSTZV

Entry points: DFHBSTZV

Called by: DFHTBSxx

Description: DFHBSTZV adds the parts of a terminal or session TCT table entry that are special to VTAM and IRC.

DFHBSTZZ

Entry points: DFHBSTZZ

Called by: DFHTBSxx

Description: DFHBSTZZ adds a non-APPC session to the TCT. (APPC is advanced program-to-program communication.)

DFHBSTZ1

Entry points: DFHBSTZ1

Called by: DFHTBSxx

Description: DFHBSTZ1 adds support for a remote terminal to a CICS system.

DFHBSTZ2

Entry points: DFHBSTZ2

Called by: DFHTBSxx

Description: DFHBSTZ2 adds support for a remote advanced program-to-program communication (APPC) connection.

DFHBSTZ3

Entry points: DFHBSTZ3

Called by: DFHTBSxx

Description: DFHBSTZ3 adds a 3270 to the TCT.

DFHBSXGS

Entry points: DFHBSXGS

Called by: DFHBSMIR, DFHZTSP

Description: DFHBSXGS generates a unique session name for an LU6.2 TCTTE.

DFHBSZZ

Entry points: DFHBSZZ

Called by: DFHTBSxx

Description: DFHBSZZ adds a terminal or session to the TCT.

DFHBSZZS

Entry points: DFHBSZZS

Called by: DFHTBSxx

Description: DFHBSZZS adds a new session to LU6.2 support.

DFHBSZZV

Entry points: DFHBSZZV

Called by: DFHTBSxx

Description: DFHBSZZV adds a VTAM terminal or session to the TCT.

DFHCAPB

Entry points: DFHCAPNA

Called by: DFHTCRP

Description: DFHCAPB processes command analysis for VTAM terminal definitions contained in a load module table DFHRDTxx for TCT migration.

DFHCCNV

Entry points: DFHCCNV

Called by: DFHCHS, DFHMIRS

Description: DFHCCNV provides conversion of user data from ASCII to EBCDIC and from EBCDIC to ASCII for function-shipped requests from CICS OS/2 users. It is called from either the LU2 remote server program DFHCHS or the mirror program DFHMIRS, for EXEC CICS requests and replies originating from the identified server or mirror. For any function-shipped request it is invoked twice, once on the inbound side and once on the outbound path. DFHCCNV is passed the EXEC CICS parameter list by its caller. On the request side, this occurs after DFHCHS or DFHMIRS has called transformer 2 but before DFHEIP is invoked. On the response side, this occurs after DFHEIP returns to DFHCHS or DFHMIRS but before transformer 3 is invoked. External reference is made to a pregenerated CICS OS/2 conversion table, DFHCNV.

DFHCMP

Entry points: DFHCMPNA

Called by: DFHETR

Description: The CICS monitoring compatibility module is invoked by the old event monitoring point of EXEC CICS ENTER TRACEID to interface to the monitoring domain.

DFHCPY

Entry points: DFHCPYNA

Called by: DFHPRK

Description: The 3270 copy program (transaction CSCY) causes data to be copied from screen to printer in a (VTAM) 3270 system. DFHCPY is invoked by DFHPRK (only if the 3270 has the copy feature) and issues a DFHTC TYPE=COPY macro to the printer. DFHCPY then initiates DFHRKB.

DFHCRC

Entry points: DFHCRCNA

Called by: MVS

Description: The interregion abnormal exit module is a CICS module that contains an ESTAE exit to terminate interregion communication in abnormal conditions. DFHCRC issues a CLEAR request to the interregion SVC.

DFHCRNP

Entry points: DFHCRNNA

Called by: DFHCRSP, dispatcher

Description: DFHCRNP, the connection manager (transaction CSNC), controls IRC connections. It establishes and breaks these connections and processes inbound requests to attach tasks (for example, mirror) to communicate with connected systems.

DFHCRQ

Entry points: DFHCRQNA

Called by: transaction CRSQ

Description: The remote schedule page program is invoked periodically to delete requests to attach a transaction on a remotely owned terminal if those requests have been outstanding for more than the ATI purge delay interval.

DFHCRR

Entry points: DFHCRRNA

Called by: DFHCRNP

Description: The interregion session recovery program performs session recovery on behalf of primary or secondary IRC sessions.

DFHCRS

Entry points: DFHCRSNA

Called by: transaction CRSR

Description: The remote scheduler program builds and ships AIDs for automatic transaction initiation when the terminal is in a remote address space. It receives requests to schedule an AID shipped to it from a remote address space.

DFHCRSP

Entry points: DFHCRSNA

Called by: DFHEIP, DFHSIJ1

Description: The interregion communication startup module can be invoked, either at system initialization or by a CEMT request, in order to make the CICS address space available for communication by other address spaces. DFHCRSP issues a logon request to the interregion communication SVC routine and attaches transaction CSNC (DFHCRNP).

DFHCRT

Entry points: DFHCRTNA

Called by: transaction CXRT

Description: DFHCRT is the relay program used when a transaction attempts to allocate a conversation to a remote advanced program-to-program (APPC) terminal.

DFHCSA

Entry points: DFHCSANA

Called by: Not applicable

Description: The DFHCSA module contains the common system area (CSA) and CSA optional features list, the queue control area (QCA) and, for HPO systems, the SRB interface control area.

DFHCSDUP

Entry points: DFHCUCNA

Called by: MVS

Description: The CSD utility program is an offline program that provides services for the CSD. The utility command processor (DFHCUCP) validates commands and invokes the appropriate routine to execute the requested function. DFHCSDUP calls DFHDMP to access the CSD.

DFHCSSC

Entry points: DFHCSSNA

Called by: DFHSIJ1, DFHSNSN, DFHSUSN, DFHTCRP, DFHZCUT

Description: DFHCSSC, the signon time-out program, is invoked as a system task by DFHSIJ1 and DFHTCRP to perform XRF takeover sign-off time-out processing. It is invoked elsewhere as the CSSC transaction for time-out processing of the following:

- Terminals signed on with the TIMEOUT option
- Entries in the internally managed signon table (SNT)
- Entries in the local userid tables (LUITs).

The CSSC transaction is scheduled when task termination determines that a time-out is necessary. When DFHCSSC is executed, it examines all signed-on terminals, all entries in the SNT managed by DFHTMP, and all entries in the LUITs. It signs off or deletes expired entries as appropriate, and then reschedules itself to perform later time-outs if required.

DFHCSVC

Entry points: DFHCSVC

Called by: MVS

Description: This module is a type 3 SVC that passes control to the various required routines, dependent on the parameter passed to it. On a first request for a particular function, it loads the required module and puts its address in the AFCB and then branches to that code. Further calls result in the address in the AFCB being branched to.

Returns to: Type 3 SVC

DFHCUCAB

Entry points: DFHCUCA

Called by: DFHCAPB

Description: The resource definition online command analyzer interprets a VTAM resource definition in command form and produces a parameter list.

DFHCUCB

Entry points: DFHCUCB

Called by: DFHCUCP

Description: The resource definition online command builder receives commands and transforms them to a format for use by the command processors.

DFHCUCCB

Entry points: DFHCUCC

Called by: DFHCAPB

Description: This program extracts a single entry from a loaded RDT table containing VTAM resource definitions for TCT migration.

DFHCUCDB

Entry points: DFHCUCD

Called by: DFHCAPB

Description: The resource definition online command default values program modifies the parameter list produced by DFHCUCAB by inserting the default values.

DFHCWTO

Entry points: DFHCWTNA

Called by: CWTO transaction

Description: The console write-to-operator module is a CICS-supplied transaction that allows an operator to send a message to the console operator. DFHCWTO issues SVC 35 (WTO) to pass the message to the operator's console.

DFHDBAT

Entry points: AENTRY

Called by: DFHERM, IMS database resource adapter (DRA).

Description: This program provides a mapping between the external architectures of CICS (the resource manager interface (RMI) and of DBCTL (the database resource adapter (DRA)). Both are independently defined and different. DFHDBAT is part of the support for the CICS-DBCTL interface and runs in an application program environment. DFHDBAT is invoked by a DFHRMCAL request through the CICS RMI. The RMI supplies DFHDBAT with a parameter list from which DFHDBAT constructs the DRA INIT, DRA TERM, and DRA THREAD parameter lists. It must also transform the DRA parameter list back, after a DL/I call to the format expected by CICS. Thus, DFHDBAT is also referred to as the CICS-DBCTL adapter-transformer.

DFHDBCON

Entry points: DFHDBCON

Called by: DFHDBME

Description: This program issues a CICS-DBCTL interface connection request to the CICS-DBCTL adapter-transformer, DFHDBAT. DFHDBCON is part of the support for the CICS-DBCTL interface and runs in an application program environment.

DFHDBCR

Entry points: DFHDBCR

Called by: DFHSII1 via attach

Description: DFHDBCR is the CICS/DBCTL XRF tracking program. DFHDBCR runs in an alternate CICS system during the tracking phase. DFHDBCR receives messages from the active CICS system regarding the state of the connection to DBCTL, and drives the XXDFB and XXDTO exits and takes appropriate action.

DFHDBCT

Entry points: DFHDBCT

Called by: DFHDBCTX, DFHDBAT

Description: This program processes any elements placed on the CICS-DBCTL control work element (CWE) chain. DFHDBCT is part of the support for the CICS-DBCTL interface and runs in an application program environment. It is invoked when the CICS-DBCTL connection program, DFHDBCON, attempts to connect to DBCTL. The program then issues a wait. The DFHDBCT program is posted whenever an element is placed on the CWE chain.

DFHDBCTX

Entry points: DFHDBCTX

Called by: DFHDBAT

Description: This program notifies the CICS-DBCTL control transaction of changes in the state of the CICS-DBCTL interface. DFHDBCTX is part of the support for the CICS-DBCTL interface. It does not run in a CICS environment and thus does not use any CICS services. This exit is invoked by the DBCTL adapter on behalf of the DBCTL DRA.

DFHDBDI

Entry points: DFHDBDI

Called by: DFHDBCT

Description: This program disables the CICS-DBCTL adapter program and cleans up the storage used by the CICS-DBCTL interface programs. DFHDBDI is part of the support for the CICS-DBCTL interface and runs in an application program environment. DFHDBDI is invoked by the CICS/VS DBCTL control program, DFHDBCT, just before it terminates.

DFHDBDSC

Entry points: DFHDBDSC

Called by: DFHDBCT, DFHDBME

Description: This program issues a CICS-DBCTL interface disconnection request to the CICS-DBCTL adapter-transformer. DFHDBDSC is part of the support for the CICS-DBCTL interface and runs in an application program environment.

DFHDBIQ

Entry points: DFHDBIQ

Called by: CDBI transaction

Description: This program is the CDBI CICS-supplied transaction. Its function is to inquire on the current status of the CICS-DBCTL interface. DFHDBIQ is part of the support for the CICS-DBCTL interface.

DFHDBME

Entry points: DFHDBME

Called by: CDBC transaction

Description: This program is the CDBC CICS-supplied transaction. Its function is to provide a front end for making certain changes to the status of the CICS-DBCTL interface. DFHDBME is part of the support for the CICS-DBCTL interface.

DFHDBMOX

Entry points: DFHDBMOX

Called by: DFHDBAT

Description: This program outputs monitoring information supplied by DBCTL to the monitoring domain, using monitoring domain services. The information is supplied by DBCTL when it has processed a PSB schedule request and a thread termination request. This exit forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment. This exit is invoked by the CICS-DBCTL adapter.

DFHDBP

Entry points: DFHDBPNA

Called by: DFHAPRC

Description: This program invokes DWE processors when a UOW backs out.

DFHDBREX

Entry points: DFHDBREX

Called by: DFHDBAT

Description: This program is the CICS-DBCTL resume exit. The resume exit is driven whenever the adapter or the DRA requires to resume a task which they have suspended. This exit forms part of the support for the CICS-DBCTL interface. It does not run in a CICS environment and thus cannot use CICS services.

DFHDBSPX

Entry points: DFHDBSPX

Called by: DFHDBAT

Description: This program is the CICS-DBCTL suspend exit. The suspend exit is driven whenever the adapter or the DRA requires to suspend a task. DFHDBSPX forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment.

DFHDBSSX

Entry points: DFHDBSSX

Called by: DFHDBAT

Description: DFHDBSSX is the CICS/DBCTL status exit. In the event of a DRA thread failure, DFHDBSSX is called to transfer ownership of PCB storage to CICS. When the task ends, DFHDBSSX is called to release this storage.

DFHDBSTX

Entry points: DFHDBSTX

Called by: DFHDBAT

Description: This program is the CICS-DBCTL statistics exit. The exit outputs CICS-DBCTL session termination statistics to the statistics domain. DFHDBSTX forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment, but it can also be invoked during CICS orderly termination. This exit is invoked by the CICS-DBCTL adapter.

DFHDBTOX

Entry points: DFHDBTOX

Called by: DFHDBAT

Description: This program is the CICS-DBCTL token exit. The function of this exit is to provide the CICS-DBCTL adapter with task tokens for tasks that have not been through the DBCTL call processor ,DFHDLIDP, or the DBCTL connection program, DFHDBCON, or the DBCTL disconnection program, DFHDBDSC, where task tokens are usually generated. DFHDBTOX forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment. This exit is invoked by the CICS-DBCTL adapter.

DFHDBUEX

Entry points: DFHDBUEX

Called by: DFHDBCT, DFHDBDSC

Description: DFHDBUEX is the user-replaceable CICS-DBCTL exit program. It is invoked whenever CICS successfully connects to DBCTL and whenever CICS disconnects from DBCTL. DFHDBUEX forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment.

DFHDCP

Entry points: DFHDCPNA

Called by: DFHDC macro, DFHEDC

Description: DFHDCP translates DFHDC macro requests for a transaction dump to DU domain TRANSACTION_DUMP calls.

DFHDES

Entry points: DFHDESNA

Called by: DFHZEV1, DFHZEV2, DFHZOPN

Description: DFHDES performs data encryption and bind-time security.

DFHDIP

Entry points: DFHDIPNA

Called by: DFHACP, DFHDI macro, DFHEDI, DFHKCP, DFHMCP, DFHTOM, DFHZEMW, DFHZRSP, DFHZSUP

Description: The data interchange program acts as a function manager when transactions want to communicate with batch devices using SNA support. DFHDIP builds and receives FMHs, which control the data set selection and function currently being performed by the batch device.

The main subroutines of DFHDIP are:

```
DESTCHEK - Destination change
D1ABORTE - Abort
D1CONRTE - Continue
D1ENDRTE - End
D1INARTE - Transaction attach
D1INPRTE - Input
D1NOTRTE - Note
D1QUERTE - Query.
```

DFHDLI

Entry points: DFHDLINA

Called by: User application, DFHMIRS, DFHSPP

Description: DFHDLI is the DL/I call router program. It decides which call processor is to be used for the request: DBCTL or REMOTE. It then invokes the appropriate processor: DFHDLIDP or DFHDLIRP.

DFHDLIAI

Entry points: ASMTDLI, CBLTDLI, PLITDLI

Called by: User application using DL/I CALL interface

Description: This module is used by the CICS-DL/I interface. It is link-edited with the application program to provide D/I communication between the application and the CICS-DL/I interface routine DFHDLI. Calls for DL/I to the ASMTDLI, CBLTDLI, or PLITDLI entry points are resolved by this processor.

DFHDLIDP

Entry points: DFHDLIDP

Called by: DFHDLI

Description: DFHDLIDP is the DBCTL call processor. It services DL/I calls for PSBs that are owned by a DBCTL subsystem, and invokes the DL/I task-related user exit (adapter) to interface with DBCTL.

DFHDLIRP

Entry points: DFHDLIRP

Called by: DFHDLI

Description: DFHDLIRP is the remote call processor. It services DL/I calls that are function-shipped to another CICS system.

DFHDMP

Entry points: DFHDMPNA

Called by: DFHAMP, DFHCSDUP

Description: The definition file management program handles physical changes to the CSD. The main processes in DFHDMP are:

```
BUILDKWA (DM16)  - Build key work area
CONNECT (DM01)   - CONNECT
CREATSET (DM11)  - Create SET
DELETE (DM05)    - DELETE
DISCONN (DM02)   - DISCONNECT
ENDBRO (DM10)    - End BROWSE
ERASESET (DM12)  - Delete SET
GETNEXT (DM09)   - Get next record
LOCK (DM06)      - LOCK
QUERYSET (DM13)  - QUERYSET
READ (DM04)      - Read CSD control records
RELSEKWA (DM17)  - Free key work area
SETBRO (DM08)    - Set browse
UNLOCK (DM06)    - UNLOCK
WRITE (DM03)     - WRITE.
```

DFHDRPG

Entry points: DFHDRPNA

Called by: DFHEIP

Description: DFHDRPG is the EXEC interface processor for EXEC DLI commands for database sharing. It receives the parameters of the command and from them builds a list that is appropriate to call DFHDRPE, the program request handler. On return from DFHDRPE, the status code in the PCB is examined. For some codes, an MVS abend is executed; the other codes are passed back to the application program.

DFHDSBA$, DFHDSB1$

Entry points: DFHDSBNA

Called by: DFHPBP

Description: The data stream build program produces the final device-dependent data stream for each page of BMS output. It is invoked only for processing data streams that are not in 3270 format. DFHDSB removes blanks from the ends of lines, converts logical new-line characters into the device-dependent equivalents (adding idle characters where necessary), and inserts horizontal and vertical tab characters if supported.

DFHDU640

Entry points: DFHDUPNA

Called by: MVS

Description: The dump utility program formats and prints transaction dumps from a CICS transaction dump data set (DFHDMPA or DFHDMPB). The transaction dumps are written to the data set by the dump domain. They contain information about the state of a particular transaction at the time of a transaction abend or user-requested dump.

DFHDXACH

Entry points: DFHDXACH

Called by: DFHDBCR, DFHDBCT

Description: DFHDXACH is a stub that is also MVS-attached, and which branches to an input address.

DFHDXSTM

Entry points: DFHDXSTM

Called by: DFHDBCT, DFHDBCR

Description: DFHDXSTM is used to attach, detach, and inquire on MVS subtasks attached by DFHDBCR and DFHDBCT.

DFHDYP

Entry points: DFHDYP

Called by: DFHAPRT

Description: This is the system-provided (default) dynamic routing program invoked from the CICS relay program (DFHAPRT) when a remote transaction is defined as being dynamic.

DFHEAI

Entry points: DFHEI1

Called by: User application

Description: This is a stub that is link-edited with an assembler-language application program to provide communication with DFHEIP. The command-language translator turns each EXEC CICS command into a call statement. The external entry point invoked by the call is resolved to an entry point in this stub. The address of the entry point in DFHEIP (DFHEIPCN) is found through a chain of system and CICS control blocks.

The DFHEAI stub must be included at the beginning of
the program in the output from the link edit. To achieve
this, ORDER and INCLUDE statements for DFHEAI
must be in the link-edit step of your JCL. When you use
the CICS-supplied assembler procedure DFHEITAL in
the SDFHPROC library to translate, assemble, and
link-edit application programs written in assembler
language, the COPYLINK step of this procedure copies
SDFHMAC(DFHEILIA). DFHEILIA contains the following
statements that must be included:

```
#      ORDER  DFHEAI
#      INCLUDE  SYSLIB(DFHEAI)
```

The statements are put into a temporary file that is
concatenated before the assembled application program
in the LKED step of the procedure.

DFHEAI0

Entry points: DFHEAI0

Called by: User application

Description: This is a stub that is link-edited with an assembler-language application program to provide communication with DFHEIPA, part of the EXEC interface layer, for the prolog and epilog calls generated by the command-language translator in the application program. The external entry point invoked by the calls is resolved to an entry point in this stub. The address of the entry point in DFHEIPA (DFHEIPAN) is found using a chain of system and CICS control blocks.

DFHEAP1$

Entry points: PREPROC

Description: The assembler-language translator module performs the following functions:

- Runs offline.
- Takes on an input file.
- Produces an output or listing file.
- Gives a return code according to the highest severity of the message produced:

```
 0 - no message
 4 - warning
 8 - error
12 - severe error
16 - translator failure.
```

- Replaces CICS commands by invocations of the DFHECALL macro, and inserts invocations of DFHEIENT, DFHEIRET, DFHEISTG, and DFHEIEND macros at appropriate places.
- Inserts diagnostics resulting from errors in commands, as comments in the output program that are not listed on the listing file.

DFHEBF

Entry points: DFHEBFNA

Called by: DFHEIP

Description: DFHEBF is the EXEC interface processor for the field edit built-in function, DEEDIT.

DFHEBU

Entry points: DFHEBUNA

Called by: DFHETL, DFHETC

Description: The EXEC function management header (FMH) construction module is called by DFHETC when a SEND or CONVERSE command is being processed, and ATTACH function management headers have to be built and concatenated ahead of user data.

DFHECI

Entry points: DFHEI1

Called by: User application

Description: This is a link-edit stub similar to DFHEAI, except that it is used for COBOL application programs.

DFHECID

Entry points: DFHEIN01

Called by: DFHECIP

Description: The command interpreter module analyzes CECI commands, and manages its displays. It uses the EXEC interface to invoke other CICS functions.

DFHECIP

Entry points: DFHEIN00

Called by: CECI transaction

Description: The command interpreter program performs preliminary validation and initialization for the CECI transaction, and links to DFHECID.

DFHECP1$

Entry points: PREPROC

Description: The COBOL translator module performs the following functions:
- Runs offline.
- Takes on an input file.
- Produces an output or listing file.
- Gives a return code according to the highest severity of the message produced:

```
 0 - no message
 4 - warning
 8 - error
12 - severe error
16 - translator failure.
```

- Inserts DFHEIBLK and COMMAREA declarations in the LINKAGE section.
- Inserts the EIB definition in the LINKAGE section.
- Inserts the DIB definition (for DL/I HLPI) in the WORKING_STORAGE section.
- In the PROCEDURE DIVISION, the translator inserts a USING clause in the DIVISION statement, and replaces all CICS and DL/I commands by COBOL CALL statements.
- Inserts diagnostics resulting from any errors in commands, as messages in the translator listing file.

DFHEDAD

Entry points: DFHESP01

Called by: DFHEDAP

Description: The resource definition online (RDO) transactions module analyzes the commands, and manages the displays for CEDA, CEDB, and CEDC. It uses the EXEC interface.

DFHEDAP

Entry points: DFHESP00

Called by: CEDA, CEDB, CEDC transaction

Description: The resource definition online (RDO) transactions program performs preliminary validation and initialization for CEDA, and links to DFHEDAD.

Returns to: DFHEIP

DFHEDC

Entry points: DFHEDCNA

Called by: DFHEIP

Description: DFHEDC is the EXEC interface processor for dump commands.

DFHEDFBR

Entry points: DFHEDFBR

Called by: CEBR transaction, DFHEDFD

Description: The temporary-storage browse transaction browses, copies, or deletes entries in a temporary-storage queue. It interprets commands and PF key actions.

DFHEDFD

Entry points: DFHEDFD

Called by: DFHEDFP

Description: The EDF display program is invoked from DFHEDFP to analyze and display the current status of the user program. DFHEDFD stores control information about a temporary-storage message queue and uses BMS to format the display screen. DFHEDFD interfaces with other CICS control programs using the EXEC interface.

DFHEDFM

Description: The EDF map set contains BMS maps used by DFHEDFD to format the EDF display.

DFHEDFP

Entry points: DFHEDFNA

Called by: transaction CEDF

Description: The EDF main program is the control program for EDF. DFHEDFP can be invoked in one of two ways:
1. Directly from the EDF display terminal by entering the CEDF transaction identification
2. By pressing the user-defined PF key.

DFHEDFP is also attached by DFHEDFX as the main program of the EDF task.

DFHEDFR

Entry points: DFHEDFNA

Called by: Not applicable

Description: The EDF response table contains a description of the exception responses for each EXEC command and the abend codes associated with error responses. DFHEDFR is used by DFHEDFD to interpret the responses obtained from an EXEC command.

DFHEDFX

Entry points: DFHEDFNA

Called by: DFHACP, DFHEIP, program manager

Description: The EDF task switch program is invoked from DFHACP, DFHEIP, or program manager when a program is running in debug mode. DFHEDFX suspends the user task and attaches the debugging task, passing it information about the user task in the TWA of the debugging task.

DFHEDI

Entry points: DFHEDINA

Called by: DFHEIP

Description: DFHEDI is the EXEC interface processor for data interchange commands.

DFHEDP

Entry points: DFHEDPNA

Called by: DFHERM

Description: DFHEDP converts command-level DL/I statements into a call parameter list acceptable to DL/I. In addition, it provides 31-bit application support by moving segment I/O areas above and below the 16MB line as required.

DFHEDP1$

Entry points: PREPROC

Description: The C translator module performs the following functions:
- Runs offline.
- Takes on an input file.
- Produces an output or listing file.
- Gives a return code according to the highest severity of the message produced:
```
 0 - no message
 4 - warning
 8 - error
12 - severe error
16 - translator failure.
```
- Inserts the EIB definition at the head of the translated output.
- If the DLI translator option is specified, inserts the DIB definition
- Replaces all CICS and DL/I commands in the input program by function calls (dfhexec) in the output program.
- Inserts diagnostics from any errors in commands, as messages on the translator listing file.

DFHEEI

Entry points: DFHEEINA

Called by: DFHEIP

Description: DFHEEI is the EXEC interface processor for DFHEIP ADDRESS, ASSIGN, PUSH, POP, and HANDLE commands.

DFHEEX

Entry points: DFHEEXNA

Called by: DFHETC

Description: The EXEC function management header (FMH) extraction module is called by DFHETC when a RECEIVE or CONVERSE command is being processed, and when data has to be extracted from ATTACH function management headers.

DFHEFRM

Entry points: DFHEFRM

Called by: DFHDBP, DFHSPP

Description: DFHEFRM is the EXEC interface file control syncpoint processor. At syncpoint commit or rollback time, DFHEFRM deletes the FFLE entries that were created by DFHFCEI for the task.

DFHEGL

Entry points: DFHEGLNA

Called by: DFHEIP

Description: DFHEGL is the EXEC interface processor for unmapped LU6.2 commands.

DFHEIIC

Entry points: DFHEICNA

Called by: DFHEIP

Description: DFHEIIC is the EXEC interface processor for interval control commands.

DFHEIDTI

Entry points: DFHEIDTI

Called by: DFHEIP

Description: DFHEIDTI is the EXEC interface processor for ASKTIME and FORMATTIME. DFHEIDTI updates the time and date fields in the EIB and certain time fields in the CSA, and returns the current time, or date, to the application.

DFHEIP

Entry points: DFHEIPNA

Called by: application programs

Description: DFHEIP is the main EXEC interface module. See Chapter 19, “EXEC interface,” on page 135 for further information.

DFHEIPA

Entry points: DFHEIPAN

Called by: DFHEAI0

Description: DFHEIPA is part of the EXEC interface layer. It acquires and partially initializes the DFHEISTG dynamic storage when called from the DFHEIENT macro in an assembler-language application program. It frees this storage when called from the DFHEIRET macro.

DFHEIFC

Entry points: DFHEIFC

Called by: DFHEIP

Description: DFHEIFC is the file control EXEC interface module, providing an interface between DFHEIP and file control. It locates the AFCTE, and performs the security check. For a remote file, DFHEIFC passes the request to a transformer, which then ships the request to the other system. For a local file, DFHEIFC converts the EXEC argument list to an FCFR parameter list (as defined by the DFHFCFRA DSECT) and calls DFHFCFR, the file control file request handler. After the request completes, DFHEIFC builds return code information in the EIB.

DFHEISR

Entry points: DFHEISR

Called by: DFHEDI, DFHEGL, DFHEIQMS, DFHEMS, DFHEOP, DFHETC, DFHETL, DFHTDB, DFHXFFC, DFHXFX

Description: DFHEISR obtains buffers and copies data for the calling EXEC interface modules, at the location and in the storage key required by the application.

DFHEJC

Entry points: DFHEJCNA

Called by: DFHEIP

Description: DFHEJC is the EXEC interface processor for journaling commands.

DFHEKC

Entry points: DFHEKCNA

Called by: DFHEIP

Description: DFHEKC is the EXEC interface processor for task control commands.

DFHELII

Entry points: DFHEI1

Called by: User application

Description: This is a link-edit stub similar to DFHEAI, except that it is used for C application programs.

DFHEMS

Entry points: DFHEMSNA

Called by: DFHEIP

Description: DFHEMS is the EXEC interface processor for BMS commands.

DFHEMTA

Entry points: DFHEMT00

Called by: User application

Description: The master terminal programmed interface program is a special version of DFHEMTP that a user application can link to for master terminal services.

DFHEMTD

Entry points: DFHEMT01

Called by: DFHEMTA, DFHEMTP, DFHEOTP, DFHESTP

Description: The master terminal module analyzes the commands, and manages displays for CEMT, CEOT, and CEST transactions. It uses the EXEC interface.

DFHEMTP

Entry points: DFHEMT00

Called by: CEMT transaction

Description: The master terminal program performs preliminary validation and initialization for the CEMT transaction, and links to DFHEMTD.

DFHEOTP

Entry points: DFHEMT00

Called by: CEOT transaction

Description: The master terminal program performs preliminary validation and initialization for the CEOT transaction, and links to DFHEMTD.

DFHEPC

Entry points: DFHEPCNA

Called by: DFHEIP

Description: DFHEPC is the EXEC interface processor for program control commands.

DFHEPI

Entry points: DFHEI1

Called by: User application

Description: This is a link-edit stub similar to DFHEAI, except that it is used for PL/I application programs.

DFHEPP1$

Entry points: PREPROC

Description: The PL/I translator module performs the following functions:
- Runs offline.
- Takes on an input file.
- Produces an output or listing file.
- Gives a return code according to the highest severity of the message produced:

```
 0 - no message
 4 - warning
 8 - error
12 - severe error
16 - translator failure.
```

- If the input program is a MAIN procedure, inserts DFHEIPTR as the first parameter on the PROCEDURE statement to address the EIB. The translator also inserts declarations of the EIB and certain temporary variables.
- Replaces all CICS and DL/I commands in the input program by CALL statements in the output program.
- Inserts diagnostics from any errors in commands, as messages on the translator listing file.

DFHEPS

Entry points: DFHEPSNA

Called by: DFHEIP

Description: DFHEPS is the link between DFHEIP and the JES interface program, DFHPSP.

DFHERM

Entry points: DFHERMNA

Called by: DFHEIP

Description: DFHERM is called by DFHEIP on behalf of the other components of CICS to manage the connection between CICS and non-CICS products.

DFHESC

Entry points: DFHESCNA

Called by: DFHEIP

Description: DFHESC is the EXEC interface processor for storage control commands.

DFHEISP

Entry points: DFHESPNA

Called by: DFHEIP

Description: DFHEISP is the EXEC interface processor for syncpoint commands.

DFHESTP

Entry points: DFHEMT00

Called by: CEST transaction

Description: The master terminal program performs preliminary validation and initialization for the CEST transaction, and links to DFHEMTD.

DFHETC

Entry points: DFHETCNA

Called by: DFHEIP

Description: DFHETC is the EXEC interface processor for terminal control commands.

DFHETD

Entry points: DFHETDNA

Called by: DFHEIP

Description: DFHETD is the EXEC interface processor for transient data commands. The EXEC requests are routed from DFHETD to DFHTDP using the corresponding DFHTD CTYPE requests.

DFHETL

Entry points: DFHETLNA

Called by: DFHETC

Description: DFHETL is the EXEC interface processor for mapped LU6.2 commands.

DFHETR

Entry points: DFHETRNA

Called by: DFHEIP

Description: DFHETR is the EXEC interface processor for trace commands.

DFHETS

Entry points: DFHETSNA

Called by: DFHEIP

Description: DFHETS is the EXEC interface processor for temporary-storage commands.

DFHEXI

Entry points: DFHEXINA

Called by: DFHZARQ

Description: The exceptional input program is invoked from DFHZCP when unexpected input is received from a VTAM 3270 terminal that has a task attached. DFHEXI checks whether the input is the result of a 3270 print function key being pressed; if so, DFHEXI issues a DFHTC TYPE=PRINT macro, and then unlocks the keyboard; in any case, DFHEXI then passes control back to DFHZCP.

DFHFCAT

Entry points: DFHFCAT

Called by: DFHFCDN, DFHFCN

Description: DFHFCAT processes inquire and update requests on the state of the backup while open (BWO) attributes in the ICF catalog for VSAM data sets, and inquires on the quiesce state in the ICF catalog.

DFHFCBD

Entry points: DFHFCBD

Called by: DFHFCFR

Description: DFHFCBD handles BDAM file control requests except for OPEN and CLOSE.

DFHFCDN

Entry points: DFHFCDN

Called by: DFHAMFC, DFHAMPFI, DFHEIQDN, DFHEIQDS, DFHFCLF, DFHFCMT, DFHFCN, DFHFCRC, DFHFCRO, DFHFCRD, DFHFCRP

Description: DFHFCDN builds data set name blocks at cold start or in response to CEDA requests. It also examines or modifies data set name blocks in response to EXEC CICS INQUIRE or EXEC CICS SET commands.

DFHFCDTS

Entry points: DFHFCDTS

Called by: DFHFCFR

Description: DFHFCDTS processes file control requests to access data table records for READ-ONLY requests against CICS-maintained tables, and for all record requests against user-maintained tables. It calls data table services to retrieve or modify table records, calls DFHFCVS to retrieve data from the VSAM source data set if it is not in the table, and calls DFHFCDTX to function ship requests that cannot be satisfied by sharing.

DFHFCFR

Entry points: DFHFCFR

Called by: DFHAPLI, DFHAPSM, DFHDTLX, DFHDMPCA, DFHEIFC, DFHERM, DFHFCDTS, DFHFCFR, DFHFCFS, DFHFCRC, DFHFCRP, DFHUEH

Description: DFHFCFR is the central module in the file control component. It handles file control requests issued by DFHFCEI (requests from application programs), or by other CICS modules (internal file control requests). DFHFCFR ensures that the file is both opened and enabled, acquires an FRTE as necessary, performs request validity checking, and then routes the request to the appropriate access-method dependent module (DFHFCBD for BDAM, DFHFCVS for non-RLS VSAM and also for update or browse requests against a CICS-maintained data table, DFHFCRS for RLS VSAM, and DFHFCDTS for all other data table requests).

DFHFCFS

Entry points: DFHFCFS

Called by: DFHAMFC, DFHDMPCA, DFHDMRM, DFHDTLX, DFHEIQDS, DFHFCDTS, DFHFCFR, DFHFCLF, DFHFCQU, DFHFCRC, DFHFCRD, DFHFCRU, DFHFCSD, DFHFCU, DFHFCVS

Description: DFHFCFS changes the state of a file. It invokes DFHFCN to open, or close, files.

DFHFCL

Entry points: DFHFCLNA

Called by: DFHFCN

Description: DFHFCL is a file control program that is link-edited into DFHFCFS. DFHFCL builds and deletes VSAM LSR pools. It is called by DFHFCN with a parameter list that specifies the pool number (1 through 8) and the action to be taken (build or delete).

DFHFCM

Entry points: DFHFCMNA

Called by: DFHFCFS

Description: DFHFCM is a file control program that is link-edited into DFHFCFS. When records are added via a VSAM path, DFHFCM is called to open the base associated with the path.

DFHFCMT

Entry points: DFHFCMT

Called by: DFHAFMT, DFHAMFC, DFHAMPFI, DFHDMPCA, DFHEDFX, DFHEIQDS

Description: DFHFCMT builds file control table entries in response to CEDA commands. It also examines or modifies FCT entries in response to EXEC CICS INQUIRE or EXEC CICS SET commands.

DFHFCN

Entry points: DFHFCNNA

Called by: DFHFCFS

Description: DFHFCN is a file control program that is link-edited into DFHFCFS. DFHFCN opens and closes files. If a file has not been allocated, DFHFCN allocates it, and frees it on closure.

DFHFCRL

Entry points: DFHFCRL

Called by: DFHAMFC

Description: DFHFCRL modifies SHRCTL blocks (describing VSAM LSR pools) in response to CEDA requests.

DFHFCRP

Entry points: DFHFCRP

Called by: DFHFCIN2

Description: The file control restart program builds the file control environment and initializes file control.

DFHFCSD

Entry points: DFHFCSD

Called by: DFHSTP

Description: DFHFCSD is called during CICS controlled shutdown processing to close all open files managed by CICS file control.

DFHFCST

Entry points: DFHFCST

Called by: DFHSTFC, DFHSTLS

Description: DFHFCST is called to collect or reset file or LSRPOOL statistics on request from DFHSTFC or DFHSTLS.

DFHFCU

Entry points: DFHFCUNA

Called by: CSFU transaction

Description: DFHFCU issues an OPEN for files specified in the file control table (FCT). This program examines the FCT, and calls DFHFCFS to open all specified files.

DFHFCVR

Entry points: DFHFCVR, UPADEXIT

Called by: DFHFCBD, DFHFCFR, DFHFCVR, DFHFCVS, VSAM

Description: DFHFCVR is a file control program that is link-edited into DFHFCVS. It handles requests to VSAM, and also contains the VSAM UPAD exit.

DFHFCVS

Entry points: DFHFCVS

Called by: DFHFCDTS, DFHFCFR

Description: DFHFCVS handles requests for file control services made against VSAM files. These services include:
- Communication with files defined in the file control table

- Logging of changes to these files by DFHFCJL and the log manager.
- Syncpoint services.

DFHFDP

Entry points: DFHFDPNA

Called by: DFHFD macro

Description: DFHFDP translates DFHFD macro requests for a system dump to DU domain SYSTEM_DUMP calls.

DFHFEP

Entry points: DFHFEPNA

Called by: CSFE transaction

Description: The FE terminal test program can be used to send a complete character set to a terminal or to echo input or to turn tracing on or off. This program is an application program and does not exit to any other CICS modules. However it does use CICS facilities.

DFHGMM

Entry points: DFHGMMNA

Called by: DFHKCP

Description: The "good morning" program is invoked by the CSGM system transaction to write a "good morning" message to VTAM logical units when a satisfactory OPNDST has occurred (and if the message has been requested in the TCT TYPE=TERMINAL entry).

DFHHPSVC

Entry points: IGCnnn

Called by: DFHZHPSR (via an SVC call)

Description: This is a type 6 SVC module used only on MVS. Its sole purpose is to cause MVS to dispatch an SRB. DFHHPSVC provides part of the CICS high performance option (HPO) code, and is invoked only if HPO is in use. In the entry point name, nnn is the number of the SVC.

Returns to: MVS

DFHICP

Entry points: DFHICPNA

Called by: DFHEIIC, DFHIC macro

Description: The interval control program is used for time management and has two main functions:
1. Services DFHIC macros under the control of a requesting task's TCA
2. Detects the expiration of time-dependent events, as defined in ICEs.

The main subroutines of DFHICP are:

```
ICCANCLN - Cancel a time-ordered request
ICEXPANL - Time expiration analysis
ICGTIMEN - Current time of day
ICGTTTDM - Data retrieval
ICICECRN - Build basic ICE
ICPCTSN  - Task initiation
ICPOSTN  - Signal expiration of a specified time
ICRESETN - Time of day clock reset support
ICSCHEDN - ICE schedule
ICWAITN  - Delay processing of a task.
```

DFHIIPA$, DFHIIP1$

Entry points: DFHIIPNA

Called by: DFHMCP

Description: The non-3270 input mapping program performs all BMS input mapping functions for all devices except the 3270. On exit from the module, the input data has been mapped into a newly acquired TIOA that is returned to the application program and is then addressable using BMS DSECTs in the application.

The main subsections of DFHIIP are:

```
IIMID  - GETMAINs TIOA to return to user, and maps
         page buffer into it using specified map.
IIREAD - Reads input data, issuing DFHTC or DFHDI
         requests to get data from the terminal.
IISCAN - Scans data stream for device-dependent
         control characters and creates page
         buffer.
```

DFHIRP

Entry points: DFHIRPNA

Called by: DFHCRC, DFHCRNP, DFHCRSP, DFHDRPD, DFHDRPE, DFHDRPF, DFHSRP, DFHSTP, DFHZCX

Description: The interregion communication program is used to pass data from one region to another in the same CEC. The programs being run in the regions are usually CICS programs, but DFHIRP does not assume this.

DFHIRW10

Entry points: As defined in interest ladder [8]

Called by: DFHIRP, DFHXMP

Description: The interregion work exit delivers work to the IRC control task (CSNC). DFHIRW10 is called whenever DFHIRP or DFHXMP has work to deliver to a system that logged on with DFHIRW10 as its interregion work exit. This module checks whether the work being delivered to the target system requires that work be enqueued on CSNC; if so, it enqueues the work and

8. **Interest ladder**: ladder within DFHIRW10 that expresses interest in all types of MRO work.

posts CSNC. DFHIRW10 is invoked in access register (AR) mode and user key.

DFHISP

Entry points: DFHISPNA

Called by: DFHDLI, DFHEIP, DFHEIFC

Description: The intersystem communication program is invoked when a request to access resource has to be shipped to a remote system (through ISC or MRO).

The requests passed to DFHISP are:
- File control
- Interval control
- Temporary storage
- Transient data
- DL/I.

DFHISP controls the acquisition, use, and freeing of a session to the remote system, and invokes DFHXFP or DFHXFX to process requests and replies. Two user exits are provided in DFHISP: XISCONA can be used to control the queuing of requests from DFHISP to allocate intersystems sessions, and XISLCLQ can be used to override the LOCALQ option of the transaction attributes. XISCONA is invoked for any function-shipping requests that cannot be processed immediately. XISLCLQ is provided to support the local queuing of function-shipped START NOCHECK requests when the link to the remote system is out of service. If a START NOCHECK request is queued, DFHISP starts the CMPX transaction when the link is brought in to service.

DFHJCP

Entry points: DFHJCPNA

Called by: DFHEJC, DFHJC macro

Description: The journal control program (DFHJCP) either processes a request to get a JCA control block, or has been called to write to a journal. In the latter case it examines the information in the JCA that is passed with the request and decides whether to call the recovery manager or the log manager based on whether it finds journalname DFHLOG in the JCA or not. There are three separate calls to the DFHLGGL gate of the log manager: one for a write, a put or a wait request. The same is true for the recovery manager calls, which use the DFHRMRE gate. In addition there is a call to this gate for requests which have keypoint record data with them.

When control returns from either of these domains, the domain's outcome is mapped onto a valid return code which is put into the JCA before control returns back to the calling program

DFHJUP

Entry points: DFHJUPNA

Called by: MVS

Description: The journal print utility program examines, selects, and displays data in QSAM data sets, such as the CICS and IMS logs. Data selection is controlled by input parameters, and an optional user exit. DFHJUP provides access to the MVS log streams via the SUBSYS keyword in the JCL.

DFHKCP

Entry points: DFHKCPNA

Called by: DFHEKC, DFHKC macro

Description: This is a startup routine that passes control to either DFHXCP or DFHXCPC. It also deals with some ENQ and DEQ calls.

DFHKCQ

Entry points: DFHKCQNA

Called by: DFHXCPC

Description: DFHKCQ processes DFHKC INITIALIZE, REPLACE, WAITINIT, and DISCARD macro calls to the transaction manager.

DFHKCRP

Entry points: DFHKCRP

Called by: DFHKCP (attaches DFHKCRP as a CICS task)

Description: DFHKCRP is the task control restart program.

DFHKCSC

Entry points: DFHKCSC

Called by: DFHKCQ

Description: This module forms part of the transaction manager. It provides the QUERY_TRANSACTION and QUERY_PROFILE functions for use in determining whether the transaction or profile specified on a DISCARD TRANSACTION or DISCARD PROFILE command respectively can validly be discarded. For the QUERY_TRANSACTION function, DFHKCSC examines the ICE chain, the AID chains, and the SIT, looking for references to the transaction that is the subject of the DISCARD. For the QUERY_PROFILE function, DFHKCSC examines the PCT for a reference to the profile that is the subject of the DISCARD.

DFHKCSP

Entry points: DFHKCSPA, DFHKCSPI, DFHKCSPD, DFHKCSPF, DFHKCSPP

Description: The task SRB control program is part of the high performance option (HPO) code available on CICS on MVS. It runs in SRB mode and resides in protected storage.

DFHLIP

Entry points: DFHLINA

Called by: DFHEDFX, DFHEIP, DFHPCPS, DFHSIJ1, DFHSTP

Description: The language interface program acts as a single point of contact between CICS and AD/Cycle Language Environment/370, and also between CICS and the language environments for VS COBOL II and C/370. To invoke a Run-Time Language Interface (RTLI) or Extended Run-Time Language Interface (ERTLI) function, the requesting module calls DFHLIP by issuing a DFHCEE FUNCTION= macro. DFHLIP performs all the interface work with the language, including the handling of any errors.

The interface functions driven by DFHLIP and the modules that call DFHLIP for those functions are as follows. An asterisk (*) after a function name shows that the function call is handled entirely within DFHLIP itself, and control remains in DFHLIP upon successful completion of the thread initialization function.

Unless otherwise indicated, each function is used for all three environments. Where alternative function names are given, the name applicable to Language Environment/370 is used in the requesting module's DFHCEE macro call regardless of the language environment.

```
DFHEDFX - Determine working storage
                  (Language Environment/370)
          OR Working storage locate
                  (VS COBOL II and C/370)
DFHEIP  - Perform GOTO
                  (Language Environment/370 only)
DFHPCPS - Establish ownership type
                  (Language Environment/370)
          OR Determine program type
                  (VS COBOL II and C/370)
        - Thread initialization
        - Run-unit initialization *
        - Run-unit begin invocation *
                  (Language Environment/370 only)
        - Run-unit end invocation
                  (Language Environment/370 only)
        - Run-unit termination
        - Thread termination
DFHSIJ1 - Partition initialization
DFHSTP  - Partition termination.
```

DFHLUP

Entry points: DFHLUPNA

Description: DFHLUP is the LU6.2 services manager. It initializes and shuts down a network, and resynchronizes flows.

DFHMCPA$, DFHMCPE$, DFHMCP1$

Entry points: DFHMCPNA

Called by: DFHBMS macro, DFHEMS

Description: The mapping control program processes DFHBMS macro requests and completes the processing of a logical message when a task terminates without issuing a DFHBMS TYPE=PAGEOUT. DFHMCP's main function is to analyze DFHBMS requests and to pass control to the appropriate modules. Other functions include the loading of maps and partition sets, and scheduling of output messages transmitted by temporary storage.

The main subsections of DFHMCP are:

```
MCPCPO   - Completes logical message build message
           control record for temporary storage
MCPDWEXT - DWE processing, invoked by DFHKCP to
           complete BMS processing at application
           termination
MCPINPT  - Handles all input requests
MCPIN    - TYPE=IN (EXEC CICS RECEIVE MAP)
MCPMAPLO - Loads map set and locates map
MCPPGBLD - TYPE=PAGEBLD|TEXTBLD (EXEC SEND TEXT)
MCPPGOUT - TYPE=PAGEOUT (EXEC CICS SEND PAGE)
MCPPURGE - TYPE=PURGE (EXEC CICS PURGE MESSAGE)
MCPROUTE - TYPE=ROUTE (EXEC CICS ROUTE).
```

DFHMCX

Entry points: DFHMCXNA

Called by: DFHMCP

Description: DFHMCX is the BMS fast path module for standard and full-function BMS, and the program for minimum BMS support. It is called by DFHMCP if the request satisfies one of the following conditions:

- It is a noncumulative direct terminal send map or receive map issued by a command-level program.
- It is for a 3270 display or an LU3 printer which does not support outboard formatting. If the terminal supports partitions, it is in the base state.
- The CSPQ transaction has been started.
- The message disposition has not changed.

DFHMGP

Entry points: DFHMGPNA

Called by: DFHACP, DFHCRQ, DFHCRT, DFHEOP, DFHFEP, DFHRTC, DFHRTE, DFHZEMW, DFHZERH, DFHZIS1, DFHZTSP, DFHZXRL

Description: The message generation program provides an interface for sending CICS messages to the terminal end user.

DFHMGT

Entry points: DFHMGTNA

Called by: DFHMGP

Description: The message prototype control table, or message generation table, consists of a series of copybooks, DFHMGTnn, each of which contains up to 100 messages that are issued by DFHMGP.

DFHMIRS

Entry points: DFHMIRNA

Called by: Task initiation

Description: The mirror program is invoked when a request to access a resource is received from a remote ISC system or from a remote MRO system. DFHMIRS may be thought of as returning the answer to the requesting actions of DFHISP. It is DFHMIRS that controls the receipt of requests and transmission of replies.

DFHMIRS processes requests from:
- MRO-connected systems
- LU6.1 connected systems
- LU6.2 sync level 1 connected systems
- LU6.2 sync level 2 connected systems.

The input to DFHMIRS consists of a TCTTE representing the session between CICS and its session partner, and a TIOA containing the function shipping request.

The TIOA is passed to DFHXFP (transformer 2) for conversion of the request from transmission format to the parameter list format required for DFHEIP or DFHDLI. If the data requires conversion (transaction CPMI), an EXEC CICS LINK is used to link to the data conversion program DFHCCNV, passing a COMMAREA that contains the EXEC CICS parameter list for the request where applicable. DFHMIRS then passes the request to DFHEIP or DFHDLI for execution.

On return from DFHEIP or DFHDLI the data conversion program is called to convert the reply (if applicable), and then the transformer program DFHXFP (transformer 3) is called to convert the reply parameter list to transmission format. DFHMIRS then determines the DFC to send with the reply and transmits the reply to the requesting system. If the mirror task has modified protected resources, it continues receiving requests and transmitting replies until a syncpoint request is received from the remote system.

A mirror task on an IRC link suspends itself on completion of a request and it is then available for use by any other MRO function-shipped request. The dispatcher terminates the mirror task if it is not reused within ten seconds.

DFHML1

Entry points: DFHML1NA

Called by: DFHMCP, DFHPBP

Description: The SCSPRT logical unit type 1 output mapping routine is called by DFHPBP to build a page of data stream from a chain of map and application data structure copies. The data contains only features that the TTP says are supported by the target terminal. This routine is called when NLEOM is specified for 3270 printers or LU3 printers.

The main subsections of DFHML1 are:

ML1SPACE
Calculate space for chaining and mapping

ML1FMCA
Format the chains that describe the maps

ML1PF Process map fields

DFHMROQP

Entry points: DFHMRONA

Called by: DFHCRNP, DFHCRSP

Description: The MRO work queue enable/disable program is invoked by the DFHMROQM macro for ENABLE and DISABLE requests (other requests are processed by an inline expansion). DFHMROQP is called by DFHCRSP to enable the MRO work queues when starting interregion communication, and by either DFHCRSP or DFHCRNP to disable the work queues when stopping interregion communication. MRO work queues are used to deliver work to the IRC control task (CSNC).

DFHMSP

Entry points: DFHMSPNA

Called by: CMSG transaction

Description: The message switching program routes a message entered at the terminal to one or more operator-defined terminals or to other operators. DFHMSP can be used in conversational mode to process operands entered from separate input operations. In this case the operands already processed are preserved in temporary storage.

The main sections and subroutines of DFHMSP are:

```
MSBMSRT  - Check for complete operands
MSCNVRS  - Issue conversational response
MSCONTIN - Process conversational response
MSMSG4   - MSG operand
MSNTRY   - Process operands
MSROUTE  - Route operand.
```

DFHMXP

Entry points: DFHMXPNA

Called by: Automatic transaction initiation

Description: The local queuing shipper provides the means of transferring to a remote system a START request that has been temporarily deferred by use of the local queuing option.

DFHM32A$, DFHM321$

Entry points: DFHM32NA

Called by: DFHMCP, DFHPBP

Description: For a BMS output request, the 3270 mapping program generates the appropriate data stream for a 3270 device, and returns control to DFHPBP which invokes the DFHTPP module to send the data to the appropriate destination, which is either to the direct terminal, or to temporary storage, or back to the caller. For a BMS input request, the data stream from a 3270 device is examined and mapped into a user application TIOA format.

The main subsections of DFHM32 are:

```
BMFMHTST - Create beginning of 3270 data stream
           (FMH cursor positioning)
BMMID    - Input mapping
BMMMS    - Merge maps (output mapping)
M32PF    - Process field.
```

DFHPBPA$, DFHPBP1$

Entry points: DFHPBPNA

Called by: DFHMCP

Description: The page and text build program positions maps or text, including header or trailer maps or text, within a page of output. For non-3270 devices, the module creates a page buffer containing the user's data which is then passed to DFHDSB to produce a device-dependent data stream. When mapping, this includes merging the data supplied by the application with the constant data included in the map. For 3270 devices, copies of the maps and application-supplied data for a page are chained together, to be processed by module DFHM32, to produce a 3270 data stream. The page and text build program creates dummy maps, and chains them in the same way for 3270 text building. For LU1 printers with extended attributes, copies of the maps and application-supplied data for a page are chained together, to be processed by module DFHML1 to produce an SCS data stream. The page and text build program creates dummy maps, and chains them in the same way for text building. After the maps have been processed by DFHDSB, DFHM32, or DFHML1, DFHPBP calls DFHTPP to write them out.

The main subroutines of DFHPBP are:

PBDOUTPT
: Mapping/text build complete, decide whether to call data stream generator and which one (DFHDSB or DFHM32). Return to caller (DFHMCP)

PBD00005
: Main control logic, request analysis.

PBD01000
: Map placement logic (3270 and non-3270 mapping).

PBD01130
: Non-3270 mapping.

PBD10000
: Pageout routine.

PBD11000
: Modify field positions within map (used by 3270 and non-3270 mapping).

PBD20000
: Text processing (3270 and non-3270).

PBD30000
: 3270 mapping.

PBFMHBLD
: Build FMH if FMHPARM specified (non-3270 text and map processing).

DFHPD640

Entry points: DFHPD640

Called by: MVS IPCS program

Description: DFHPD640 runs as an exit from the MVS IPCS program. It formats an MVS system dump (SDUMP) using the IPCS service routines to extract data and print output, including interpreted trace.

DFHPEP

Entry points: DFHPEPNA

Called by: DFHACP

Description: The program error program is CICS-supplied and establishes a base register, establishes addressability to the COMMAREA passed from DFHACP using a DFHPC CTYPE=LINK_URM macro call, and returns control to DFHACP. DFHPEP can be modified by the user to perform further recovery operations.

DFHPHP

Entry points: DFHPHPNA

Called by: DFHMCP, DFHTOM

Description: The partition handling program has one entry point, and starts with a branch table that passes control to the required routine according to the request.

The main routines of DFHPHP are:

```
PHPPSI - Loads a partition set
PHPPSC - Destroys any existing partitions and
         creates new partitions
PHPPIN - Extracts the AID, cursor position, and
```

```
        partition ID
PHPPXE - Activates the appropriate partition if
        data is received from an unexpected
        partition.
```

DFHPL1OI

Description: The PL/I interface module contains the following routines:

DFHPL1N
: Initial entry point for PL/I programs under CICS

DFHPL1I
: CICS macro service interface

DFHPL1C
: Set the CSA address

IBMBOCLA/B/C
: Startup routines for open/close functions.

DFHPRK

Entry points: DFHPRKNA

Called by: DFHZATT

Description: The 3270 print key program (transaction CSPK) is invoked when, under VTAM, the 3270 program access key designated as the print key is pressed and no task is attached to the terminal. If the 3270 hardware copy feature is present, DFHPRK attaches task CSCY to the printer designated in the TCTTE, and DFHCPY is executed. If the copy feature is not present, DFHPRK executes a DFHTC TYPE=PRINT macro.

DFHPSP

Entry points: DFHPSPNA

Called by: DFHEPS

Description: DFHPSP is the system spooling interface control module.

DFHPSPDW

Entry points: DFHPSPDW

Called by: DFHSPP

Description: DFHPSPDW is the system spooling interface DWE.

DFHPSPSS

Entry points: DFHPSPSS

Called by: DFHPSP

Description: The system spooling JES interface subtask module attaches a subtask to check whether a writer name and a token have been supplied. It opens and closes JES data sets, reads a record, and writes a record.

DFHPSPST

Entry points: DFHPSPST

Called by: DFHPSPSS

Description: DFHPSPST is the system spooling JES interface control module.

DFHPSSVC

Entry points: DFHPSSNA

Called by: DFHPSPSS, DFHPSPST

Description: DFHPSSVC is the system spooling interface module that retrieves a data set name for a given external writer name, dynamically allocates it, and returns its DDNAME.

DFHPUP

Entry points: DFHPUPNA

Called by: DFHAMP, DFHCSDUP

Description: The parameter utility program transforms the definition data of the CSD. In the CSD, the data is held in a compacted form and each field is self-identifying. Elsewhere in the processing, these fields are handled in parameterized form, using an argument address list. It also serves to transform the resource definition to the original high-level command.

DFHP3270

Entry points: DFHP32NA

Called by: CSPP transaction, DFHTCP, DFHZCP

Description: The 3270 print program prints 3270 data received from a screen on a 3270 printer. The data is compressed where possible and then transmitted to the printer.

DFHQRY

Entry points: DFHQRY

Called by: DFHALP, DFHTCTI, DFHZATT

Description: The query transaction (DFHQRY) sends a READ PARTITION QUERY structured field to a 3270, analyzes the response, and completes information in the corresponding TCTTE. DFHQRY can be attached by DFHALP, DFHTCTI, or DFHZATT.

DFHRCEX

Entry points: DFHRCEX

Called by: DFHFCBP, DFHTCBP, DFHUSBP

Description: DFHRCEX enables the global user exits for emergency restart processing.

DFHRKB

Entry points: DFHRKBNA

Called by: DFHCPY

Description: The release 3270 keyboard program is initiated by DFHCPY to release a 3270 keyboard. It does this by issuing a DFHTC TYPE=WRITE macro that sends a 3270 write control character.

DFHREST

Entry points: DFHREST

Called by: DFHXMTA

Description: The transaction restart program, DFHREST, is a user-replaceable module that helps you to determine whether or not a transaction is restarted. The default DFHREST module requests a transaction restart under certain conditions; for example, for a program isolation deadlock, one of the tasks is backed out and automatically restarted, and the other is allowed to complete its update.

DFHRLRA$, DFHRLR1$

Entry points: DFHRLRNA

Called by: DFHMCP

Description: The route list resolution program builds a terminal type parameter (TTP) control block for each type of terminal for which a message is to be built. A TTP is acquired for each terminal type in the user route list and the direct terminal if there is one.

The main subsections of DFHRLR are:

```
RLRALL   - Routing with ROUTE=ALL specified in
           application
RLRLIST  - Routing with route list specified in
           application
RLROPCL  - Routing with OPCLASS= specified in
           application
RLRRTEBY - Nonrouting, non-LDC device (that is
           direct terminal)
RLR3601  - Nonrouting LDC device.
```

DFHRMSY

Entry points: DFHRMSNA

Called by: DFHERMSP, DFHERMRS

Description: The purpose of task-related user exit resynchronization is to resolve any in-doubt LUWs. Task-related user exit resynchronization is called by DFHERMRS during execution of the RESYNC command to restore the CICS end of the thread that was interrupted by the failure of the connection with the resource manager.

It is also called by DFHERMSP when a wait is unshunted and requires RMI resynchronization with a resource manager.

DFHRTC

Entry points: DFHRTCNA

Called by: CSSF transaction

Description: The CSSF transaction is invoked on the remote system when a CRTE routing session is to be canceled. CSSF runs the CRTE cancel command processor, DFHRTC, to sign off the user and terminate the extended routing session. DFHRTC calls DFHSUSN to sign off the surrogate.

DFHRTE

Entry points: DFHRTENA

Called by: transaction CRTE, DFHSNTU

Description: The transaction routing program establishes a transaction routing session with a remote region specified by the user. Subsequent input is analyzed by DFHRTE, the transaction code extracted, and a request issued to DFHZTSP to route the transaction to the required system.

DFHSFP

Entry points: DFHSFP

Called by: CESF trans.

Description: The sign-off program signs off the user who invoked the CESF transaction.

DFHSIA1

Entry points: DFHSIANA

Called by: DFHAPSIP

Description: The DFHSIA1 system initialization program loads and initializes the CSA.

DFHSIB1

Entry points: DFHSIBNA

Called by: DFHAPSIP

Description: The DFHSIB1 system initialization program loads the CICS nucleus.

DFHSIC1

Entry points: DFHSICNA

Called by: DFHAPSIP

Description: The DFHSIC1 system initialization program initializes the transaction manager and the storage manager domain’s macro compatibility interface, acquires a TCA for LIFO functions during initialization, initializes user exits, and processes the START parameter.

DFHSID1

Entry points: DFHSIDNA

Called by: DFHAPSIP

Description: The DFHSID1 system initialization program performs the following functions:
- Adds storage subpools for transient data use
- Allocates storage for transient data control blocks:
  - TDST
  - MBCA, MBCBs, and MQCBs, I/O buffers if required
  - MRCA, ACBs, MRCBs, and RPLs
- Creates the DCTE and SDSCI for CXRF.

DFHSIF1

Entry points: DFHSIFNA

Called by: DFHAPSIP

Description: The DFHSIF1 system initialization program initializes terminal control. DFHSIF1:
- Opens the VTAM ACB
- Builds hash-table entries for non-VTAM terminals
- Constructs a DFHZCP module list in the TCT prefix
- Initializes the attach tables.

DFHSIG1

Entry points: DFHSIGNA

Called by: DFHAPSIP

Description: The DFHSIG1 system initialization program opens the dump data set.

DFHSIH1

Entry points: DFHSIHNA

Called by: DFHAPSIP

Description: The DFHSIH1 system initialization program:
- Loads the DBCTL call processor (DFHDLIDP)
- Loads the remote DBCTL call processor (DFHDLIRP) if necessary
- Attaches the TCP task.

DFHSII1

Entry points: DFHSIINA

Called by: DFHAPSIP

Description: The DFHSII1 system initialization program establishes AP domain recovery routines in DFHSRP and calls DFHICRC to initialise Interval Control services. It attaches the CPLT transaction to run the first stage PLTPI programs, the CSTP transaction (the TCP task) and a system transaction to run the rest of AP initialization (the III task). The rest of DFHSII1, running as the III task:
- Starts XRF control transactions if required
- Attaches the CICS restart tasks to run in parallel:
  - Security interface
  - Transient data
  - Terminal control
  - Program control
  - Task control
  - File control
  - Common programming interface (CPI)
  - Partner resource manager
  - Object recovery
  - Autoinstall terminal model manager
- Waits for the restart tasks to complete
- Processes the GRPLIST parameter

DFHSIJ1

Entry points: DFHSIJNA

Called by: DFHAPSIP

Description: DFHSIJ1 is the last to be executed in the process of system initialization. It issues the message 'CONTROL IS BEING GIVEN TO CICS' and passes control back to DFHAPSIP. DFHSIJ1:
- Links to DFHCRSP, if IRCSTRT=YES is specified as a system initialization parameter, to start up the interregion communication session
- Links to DFHPSIP to enable the system spooling interface
- Enables the DL/I high-level programming interface by acquiring an exit program block and addressing DFHEDP
- Enables AUTOINSTALL
- Links to the second-stage PLT programs listed in DFHPLT, then deletes this table
- Issues a DFHLDLDM SET_OPTIONS call to instruct the loader domain to write all outstanding program definitions to the catalogs.

DFHSIP

Entry points: DFHKESIP

Called by: MVS

Description: DFHSIP initializes CICS and also contains code for the following domains:
- Kernel (KE)
- Domain manager (DM)
- Dispatcher (DS)
- Dump (DU)
- Global catalog (GC)
- Local catalog (LC)
- Loader (LD)
- Lock manager (LM)
- Message (ME)
- Parameter manager (PA)
- Storage manager (SM)
- Trace (TR).

DFHSKP

Entry points: DFHSKMNA, DFHSKC, DFHSKE

Called by: MVS, DFHFCL, DFHFCM, DFHFCN, DFHPSPSS, DFHSTP, DFHXSMX

Description: DFHSKP consists of these modules, which are link-edited together:

```
DFHSKM - subtask manager
DFHSKC - subtask control program
DFHSKE - subtask execution program.
```

DFHSKM calls and, if necessary, attaches DFHSKC to process the created work queue element (WQE). DFHSKM also causes termination of the subtask when requested, and handles DWE processing and task cancel requests. DFHSKC starts an operating system subtask, DFHSKE, and waits for its completion. DFHSKE processes WQEs, looking at in-progress and waiting queues on a first-in, first-out basis. DFHSKE intercepts program checks and operating system abends.

DFHSMSCP

Entry points: DFHSMSCP

Called by: DFHSC macro

Description: The storage control program is called as a result of DFHSC GETMAIN and FREEMAIN macro requests issued from CICS modules.

DFHSNAT

Entry points: DFHSNAT

Called by: DFHCRNP, DFHZISP, DFHZSUP (via DFHSUSN)

Description: The attach-time signon/sign off interface program provides support for the signon and sign off of LU6.2 sessions.

DFHSNNFY

Entry points: DFHSNNFY

Called by: IRRDPR10

Description: The CICS segment notify exit is called by RACF whenever a change is made to a user's CICS segment in the RACF database.

DFHSNMIG

Entry points: DFHSNMIG

Called by: MVS

Description: The signon table migration utility program produces a CLIST file containing ADDUSER and ALTUSER commands that provide RACF with all the user attributes for each user entry in the signon table (SNT). This CLIST file is run by a TSO user to migrate the user information to RACF.

DFHSNP

Entry points: DFHSNP

Called by: CESN transaction

Description: The signon program is called in response to a CESN signon request. DFHSNP interprets the signon parameters, prompts the operator for more parameters if needed, and passes the values to the security manager for verification.

DFHSNSN

Entry points: DFHSNSN

Called by: DFHCSSC, DFHSNAT (via DFHSUSN)

Description: The optimized signon/sign off interface program provides a mechanism for optimizing calls to the security manager. It achieves this optimization using the signon table (SNT).

DFHSNVCL

Entry points: DFHSNVCL

Called by: IRRDPR02

Description: The OPCLASS validation exit is called by RACF to validate the operands of the OPCLASS subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands. DFHSNVCL checks whether the operands are in the range 1 through 24.

DFHSNVID

Entry points: DFHSNVID

Called by: IRRDPR02

Description: The OPIDENT validation exit is called by RACF to validate the operand of the OPIDENT subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands.

DFHSNVPR

Entry points: DFHSNVPR

Called by: IRRDPR02

Description: The OPPRTY validation exit is called by RACF to validate the operand of the OPPRTY subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands. DFHSNVPR checks whether the operand is in the range 0 through 255.

DFHSNVTO

Entry points: DFHSNVTO

Called by: IRRDPR02

Description: The TIMEOUT validation exit is called by RACF to validate the operand of the TIMEOUT subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands. DFHSNVTO checks whether the operand is in the range 1 through 60.

DFHSPP

Entry points: DFHSPPNA

Called by: DFHESP, DFHSP macro

Description: The syncpoint program is invoked during a user-specified syncpoint (by a DFHSP macro) or at task termination. For a rollback request only, DFHSPP calls DFHDBP to restore recoverable resources. It scans the DWE chain invoking the appropriate DWE processors, and performs the necessary syncpoint logging. It dequeues all resources enqueued by the transaction. DFHSPP processes any DWEs connected with the resource manager, and processes the RESYNC command.

The main subroutines of DFHSPP are:

```
SPP00005 - Write DWE log data
SPP02020 - Build a DWE chain that can be logged
SPP03000 - End.
```

DFHSRLI

Entry points: DFHSRLI

Called by: DFHSRP

Description: DFHSRLI is called during recovery processing after a system abend has occurred, to build the SRP_ERROR_DATA block and pass control to the XSRAB global user exit.

DFHSRP

Entry points: DFHSRPNA

Called by: AP domain recovery routines

Description: The system recovery program deals with program check interrupts, system abends, and runaway tasks in the AP domain. For a program check, DFHSRP abends the task with abend code ASRA. For a system abend, DFHSRP searches the SRT for the abend code that has arisen and, if a match is found, calls DFHSRLI to invoke the XSRAB global user exit (if active). Afterwards, DFHSRP can either abend CICS or attempt to keep it running with only the faulty task abended (ASRB). For a runaway task, DFHSRP abends the task with abend code AICA.

DFHSSEN

Entry points: DFHSSEN

Called by: MVS subsystem interface

Description: The subsystem end-of-memory routine is invoked by the MVS subsystem interface at all end-of-task (EOT) and end-of-memory (EOM) events when the CICS subsystem has been initialized by module DFHSSIN. It cleans up any subsystem control blocks owned by the terminating CICS region.

DFHSSGC

Entry points: DFHSSGC

Called by: DFHCSVC, DFHSSEN (through the subsystem interface)

Description: The subsystem generic connect routine records the existence of active CICS address spaces. When the first CICS address space becomes active in an MVS image, DFHSSGC enables the subsystem broadcast facility of MVS console management. When the last CICS address space becomes inactive in an MVS image, it disables the broadcast facility.

DFHSSIN

Entry points: DFHSSIN

Called by: MVS master scheduler initialization

Description: The CICS subsystem initialization routine reads subsystem parameters from SYS1.PARMLIB, and creates a subsystem vector table (SSVT) for the CICS subsystem. DFHSSIN loads modules DFHSSEN, DFHSSGC, and DFHSSWT into MVS common storage, and saves their addresses in the SSVT.

DFHSSMGP

Entry points: DFHSSMGP

Called by: DFHSSIN

Description: The subsystem interface message program provides message formatting support for the subsystem interface routines, analogous to DFHMGP within CICS. (Neither DFHMGP nor the message domain can be used in this environment because CICS is not active.)

DFHSSMGT

Entry points: DFHSSMNA

Called by: DFHSSMGP

Description: The subsystem interface message table contains the text of messages that are issued by DFHSSMGP.

DFHSSWT

Entry points: DFHSSWTA

Called by: MVS console support

Description: The subsystem interface WTO router is invoked for all MVS console messages when the console message broadcast facility has been enabled by DFHSSGC. DFHSSWT routes DFH messages to DFHSSWTO, and routes MODIFY command text to DFHSSWTF.

DFHSSWTF

Entry points: DFHSSWTF

Called by: DFHSSWT

Description: This module suppresses signon passwords that are supplied on CESN transactions entered through MODIFY commands on an MVS console. Any passwords are replaced by eight asterisks.

DFHSSWTO

Entry points: DFHSSWTO

Called by: DFHSSWT

Description: This module inserts the CICS region's applid into all DFH messages issued under a CICS TCB whose applid can be determined.

DFHSTDT

Entry points: DFHSTDT

Called by: DFHAPST

Description: This module is called by DFHAPST to collect or reset dynamic transaction backout statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTFC

Entry points: DFHSTFC

Called by: DFHAPST

Description: This module is called by DFHAPST to collect or reset file control statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTIB

Entry points: DFHSTIB

Called by: DFHAPST

Description: This module and called by DFHAPST to collect or reset IRC batch system connected statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTJC

Entry points: DFHSTJC

Called by: DFHAPST

Description: This module is called by DFHAPST to collect or reset journal control statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTLK

Entry points: DFHSTLK

Called by: DFHAPST

Description: This module is called by DFHAPST to collect or reset ISC/IRC statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTLS

Entry points: DFHSTLS

Called by: DFHAPST

Description: This module is called by DFHAPST to collect or reset LSRPOOL statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTP

Entry points: DFHSTPNA

Called by: DFHEMTP

Description: The main function of the system termination program is to shut down CICS. In sequence, DFHSTP performs the following functions (according to options specified):

1. Collects statistics now if immediate shutdown
2. Shuts down the resource managers
3. Terminates subsystem interface
4. Resumes suspended tasks
5. Executes the programs defined in the first part of DFHPLT
6. Rebuilds AIDs for paging sessions
7. Breaks the ICE and AID chains
8. Quiesces IRC
9. Executes the programs defined in the second part of DFHPLT
10. Closes all open files managed by CICS file control
11. Synchronize with Recovery Manager shutdown keypoint
12. Call WKP to catalog terminals and profiles
13. Terminate extra partition TD
14. Signs off from the CAVM
15. Terminates general-purpose subtasking facility
16. Calls the kernel to terminate the system.

Returns to: MVS

DFHSTSZ

Entry points: DFHSTSZ

Called by: DFHAPST

Description: DFHSTSZ is called by DFHAPST to collect or reset FEPI statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTTD

Entry points: DFHSTTD

Called by: DFHAPST

Description: DFHSTTD is called by DFHAPST to collect or reset transient data statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTTM

Entry points: DFHSTTM

Called by: DFHAPST

Description: DFHSTTM is called by DFHAPST to collect or reset table manager statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTTR

Entry points: DFHSTTR

Called by: DFHAPST

Description: DFHSTTR is called by DFHAPST to collect or reset terminal statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTTS

Entry points: DFHSTTS

Called by: DFHAPST

Description: DFHSTTS is called by DFHAPST to collect or reset temporary-storage statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSUSN

Entry points: DFHSUSN

Called by: DFHACP, DFHBSTS, DFHCRNP, DFHCSSC, DFHEEI, DFHEIQST, DFHERM, DFHESN, DFHMGPME, DFHMGP00, DFHRTC, DFHSUSX, DFHTCTI, DFHTPQ, DFHTPR, DFHXSMN, DFHZCUT, DFHZEV1, DFHZEV2, DFHZISP, DFHZIS2, DFHZNAC, DFHZOPN, DFHZSUP, DFHZTSP, DFHZXCU

Description: DFHSUSN is used to create, destroy, and query the contents of a signon table element (SNTTE). It calls DFHSUSX to notify the XRF alternate system of the creation and destruction of SNTTEs. It also provides an interface for the creation and validation of encrypted passwords used in LU6.2 bind password processing.

DFHSUSX

Entry points: DFHSUSX

Called by: DFHTCRPU, DFHZXCU, DFHSUSN

Description: DFHSUSX provides tracking for SNTTEs. This module is responsible for:
- Sending messages to an alternate system to reflect the current state of the SNTTEs in the active system
- Actioning an add or delete of an SNTTE in an alternate system, based on information tracked from another CICS system
- Making changes to the signed-on state in an alternate system, based on information tracked from another CICS system.

Entry points: DFHSUWT

Called by: DFHMEME, DFHSUWT

Description: The DFHSUWT module provides the following support for executing MVS WTO and WTOR SVCs:

- SEND support for Write To Operator (WTO)
- CONVERSE support for Write To Operator With Reply (WTOR).

For further information about DFHSUWT, see Chapter 68, “WTO and WTOR,” on page 503.

DFHSUZX

Entry points: DFHSUZX

Called by: DFHBSTZV, DFHEIQSC, DFHEIQST, DFHEIQTR

Description: The ZC trace controller is responsible for actioning set, cancel, and inquire requests for the CICS VTAM exit tracing facility. It sets or unsets the control flags and gets or releases the storage used by the DFHZETR function located in the ACB and RPL exits.

DFHTACP

Entry points: DFHTACNA

Called by: DFHTCP

Description: The terminal abnormal condition program is invoked by DFHTCP and performs the following functions:
- Analyzes error codes in the TACLE

- Sends appropriate messages to the CSMT transient data destination (for terminal errors), or to the CSTL transient data destination (for logical errors)
- Invokes the user-supplied (or sample) terminal error program (DFHTEP)
- Takes the appropriate actions resulting from the defaults which may have been modified by the terminal error program.

DFHTAJP

Entry points: DFHTAJNA

Description: The time adjustment program calls DFHICP to reset the CSA's time fields according to the host-supplied time-of-day. DFHTAJP then scans the ICE chain and adjusts the expiry time of interval-controlled ICEs. Time-controlled ICEs are not adjusted but the ICE chain is reordered so that it is left in order by expiry time. Times held in the TCT and CSATCNDT are decreased, and negative times are made zero. Lastly, DFHTAJP writes a message.

DFHTBSB

Entry points: DFHTBSB

Called by: DFHZCQIS

Description: DFHTBSB adds a node to the control-block structure. It is called during the dynamic installation of TCT resources, and calls routines in the control block builder.

DFHTBSBP

Entry points: DFHTBSBP

Called by: DFHTBSB, DFHTBSBP

Description: DFHTBSBP is the recursive part of DFHTBSB.

DFHTBSD

Entry points: DFHTBSD

Called by: DFHZCQDL

Description: DFHTBSD deletes a node in a CICS terminal network.

DFHTBSDP

Entry points: DFHTBSDP

Called by: DFHTBSD, DFHTBSDP

Description: DFHTBSDP is the recursive part of DFHTBSD.

DFHTBSL

Entry points: DFHTBSL

Called by: DFHTBSR, DFHZCQCH

Description: DFHTBSL creates the recovery record for a node during the dynamic installation of a TCT table entry using the CEDA INSTALL command, for example, and calls routines in the control-block builder.

DFHTBSLP

Entry points: DFHTBSLP

Called by: DFHTBSL, DFHTBSLP, DFHTBSSP

Description: DFHTBSLP is the recursive part of DFHTBSL.

DFHTBSQ

Entry points: DFHTBSQ

Called by: DFHZCQIQ

Description: DFHTBSQ is called to retrieve the parameters that were supplied to a TCT table entry at build time.

DFHTBSQP

Entry points: DFHTBSQP

Called by: DFHTBSQ

Description: DFHTBSQP is called by DFHTBSQ to retrieve parameters that were supplied to a TCT table entry at build time.

DFHTBSR

Entry points: DFHTBSR

Called by: DFHZCQRS

Description: DFHTBSR takes a table-builder recovery record and re-creates the corresponding table entry. It is called during warm or emergency restart.

DFHTBSRP

Entry points: DFHTBSRP

Called by: DFHTBSR

Description: DFHTBSRP is called by DFHTBSR.

DFHTBSSP

Entry points: DFHTBSSP

Description: DFHTBSSP performs a commit or rollback action for a previous table-builder change according to the outcome of a logical unit of work. Each action is dequeued from a DWE.

DFHTBS00

Entry points: DFHTBS

Description: DFHTBS00 is the main routine for DFHTBS and holds the addresses of the modules used to build control blocks for the dynamic installation of TCT resources.

DFHTCBP

Entry points: DFHTCBNA

Description: The terminal control backout program restores TCTTEs and other ISC state data during emergency restart.

DFHTCP

Entry points: DFHTCPNA

Description: DFHTCP is the terminal control program. The terminal control task is attached during system initialization and remains until termination. DFHTCP manages all non-VTAM terminals, which involves:
- Ensuring that I/O operations are started when possible on the lines
- Analyzing completion information
- Attaching transactions when data is received from a terminal and no task is attached to that terminal
- Servicing terminal control requests from user transactions.

The modules and subsections of DFHTCP are:

DFHTCAM
: Terminal control TCAM device dependent

DFHTCCLC
: Terminal control line control scan routine

DFHTCCOM
: Terminal control common logic

DFHTCCSS
: Terminal control start-stop common logic

DFHTCDEF
: Terminal control symbol definition

DFHTCORS
: Terminal control storage handling

DFHTCSAM
: Terminal control sequential terminal device dependent

DFHTCTI
: Terminal control task initiation

DFHTCTRN
: Terminal control translate tables

DFHTCRP

Entry points: DFHTCRP

Description: DFHTCRP initializes and recovers terminal control definitions and protected messages. It is run as a task during CICS initialization.

DFHTCRPC

Entry points: DFHTCRPC

Called by: DFHZXQO

Description: DFHTCRPC is the XRF tracking interface for TCT contents. It is one of a set of routines called by DFHZXQO from the same CALL statement, the entry point address having been passed to DFHZXQO. This routine calls ZC RESTORE to add or delete a TCT entry based on information from another CICS system using the log, the catalog, or the XRF tracking queues.

DFHTCRPL

Entry points: DFHTCRPL

Called by: DFHTCRP

Description: DFHTCRPL installs TCT resources defined by the TCT macros.

DFHTCRPS

Entry points: DFHTCRPS

Called by: DFHZXQO

Description: DFHTCRPS is the XRF tracking interface for ZCP sessions. It is one of a set of routines called by DFHZXQO from the same CALL statement, the entry point address having been passed to DFHZXQO. This routine calls DFHZXST (through DFHZXS) to make changes to the session state.

DFHTCRPU

Entry points: DFHTCRPU

Called by: DFHZXQO

Description: DFHTCRPU is the XRF tracking interface for signon table elements (SNTTEs). It is one of a set of routines called by DFHZXQO from the same CALL statement, the entry point address having been passed to DFHZXQO. This routine calls DFHSUSX to add or delete tracked SNTTEs, and to make changes to the signed-on state.

DFHTDA

Entry points: DFHTDANA

Called by: DFHAKP, DFHAMCSD, DFHAPTD, DFHCRNP, DFHCRQ, DFHDBP, DFHEIQMS, DFHEIQSQ, DFHESE, DFHETD, DFHJCP, DFHMCP, DFHMGP00, DFHRCRP, DFHRUP, DFHSII1, DFHSTP, DFHSTTD, DFHTCAP, DFHTDRP, DFHTEPM, DFHTPQ, DFHTRP, DFHTSRP, DFHWKP, DFHZNAC

Description: DFHTDA, which is link-edited with RMODE(24), handles DFHTD macro requests. In particular:

- DFHTD TYPE=GET|PUT|PURGE requests are converted to the corresponding DFHTD CTYPE=GET|PUT|PURGE requests.
- DFHTD CTYPE=GET|PUT|PURGE requests for intrapartition queues are routed to DFHTDQ for further processing.
- All of the processing for DFHTD CTYPE=GET|PUT for extrapartition queues is done under the QR TCB.
- Much of the processing for DFHTD CTYPE=OPEN|CLOSE for extrapartition queues is done under the RO TCB.

CICS Transaction Server for z/OS, Version 3 Release 1 uses QSAM GL|PL mode processing, unlike previous CICS releases which used QSAM GL|PM mode processing.

DFHTDB

Entry points: DFHTDBNA

Called by: DFHTDA

Description: DFHTDB, which is link-edited with RMODE(ANY), handles DFHTD macro requests for intrapartition queues. In particular, DFHTDB:
- Manages the input and output cursors for each queue
- Manages space on the intrapartition data set
- Initiates transactions when trigger levels are reached
- Manages the buffers; processing is done under the QR TCB
- Manages the strings; processing is done under the CO TCB.

DFHTDEXL

Entry points: EX11RTNE

Called by: QSAM

Description: DFHTDEXL contains the DCB abend exit routine used for extrapartition processing.

DFHTDP

Entry points: DFHTDANA

Called by: DFHAKP, DFHAMCSD, DFHAPTD, DFHCRNP, DFHCRQ, DFHDBP, DFHEIQMS, DFHEIQSQ, DFHESE, DFHETD, DFHMCP, DFHMGP00, DFHRCRP, DFHRUP, DFHSII1, DFHSTP, DFHSTTD, DFHTACP, DFHTDRP, DFHTEPM, DFHTPQ, DFHTRP, DFHTSRP, DFHWKP, DFHZNAC

Description: DFHTDP is a load module link-edited from object modules for DFHTDA, DFHTDEXL, and DFHTDX.

DFHTDQ

Entry points: DFHTDBNA

Called by: DFHTDA

Description: DFHTDQ is a load module link-edited from object modules for DFHTDB.

DFHTDRM

Entry points: DFHTDRM

Called by: DFHDBP

Description: DFHTDRM is the transient data recovery manager processor. If transient data has any outstanding resources, DFHTDRM is called at phase 1 syncpoint (or backout). For phase 1 syncpoint (or backout) requests, DFHTDRM issues a request to mainline transient data(DFHTDA) to reset any resources that have not yet been released.

DFHTDRP

Entry points: DFHTDRNA

Called by: DFHTDX

Description: DFHTDRP handles transient data recovery during CICS initialization. In particular, DFHTDRP:
- Adds the entries found in the DCT load module by calling the DFHTDTM gate.
- Restores input and output cursors for intrapartition queues on warm start; the cursors are recovered by DFHRUP on emergency restart
- Restores the CI state map on warm start
- Opens extrapartition queues
- Opens the intrapartition data set
- Recovers the CI state map on emergency restart.

DFHTDTM

Entry points: DFHTDTM

Called by: DFHALP, DFHEIQMS, DFHEIQSQ, DFHESE, DFHSZRPM, DFHTDRP

Description: DFHTDTM manages the entries in the destination control table. It is used to add, update and delete entries in this table and records images of each entry on the global catalog for use during a warm start or emergency restart. It allows table entries to be inquired upon.

DFHTDX

Entry points: DFHTDXNA

Called by: Task initiation

Description: DFHTDX is the initial program invoked by the transient data recovery task. It links to program DFHTDRP.

DFHTEP

Entry points: DFHTEPNA

Called by: DFHTACP

Description: The terminal error program is invoked by DFHTACP using a DFHPC CTYPE=LINK_URM macro. The sample DFHTEP (invoked only if there is no customer-supplied version) puts a terminal out of service if the number of terminal errors detected by DFHTACP exceeds default values contained in DFHTEP tables.

DFHTMP

Entry points: DFHTMPNA

Called by: DFHTM macro

Description: The table management program performs locates, adds, deletes, locks, and unlocks to entries in certain CICS tables. DFHTMP uses a hash table for these operations.

The main subroutines of DFHTMP are:

```
CHKTTC     - Check table type code
COMMIT     - Commit table changes
CRTCLE     - Create a change list element
CRTDWE     - Create deferred work element
DELDWE     - Cancel deferred work element
DEQALLDE   - Dequeue on directory element
DEQUEUE    - Dequeue on table modification
DYNHASH    - Dynamic re-hash
ENQDEQDE   - Enqueue/dequeue on directory element
ENQUEUE    - Enqueue on table modification
GET_STORAGE - Get storage from the CICS shared subpool
GET_TASK_STORAGE - Get task lifetime 31-bit storage
GET_TASK_STORAGE_COND - Get task lifetime 31-bit storage
                        (conditionally)
GET_STORAGE_FAILURE - Get storage failure routine
FREE_STORAGE - Release storage from the CICS shared subpool
FREE_TASK_STORAGE - Release task lifetime 31-bit storage
LOCATE_PREVIOUS_DE - Locate previous directory
             element in collating series
LOCATETE   - Locate a table/directory entry
LOCFDIRE   - Locate a free directory element
NOTERL     - Note Read Lock
SETABORD   - Set up alphabetic ordering pointer
             for a given table type
TMFINDLOCK - Find a read lock
TMPDWEEP   - Deferred work element processor
TMSETLOCK  - Set a read lock
TMUNLOCK   - Release a read lock
UNQUIES    - Unquiesce a directory element.
```

DFHTON

Entry points: DFHTONNA

Called by: DFHDBP, DFHSPP

Description: The terminal object resolution module is called by DFHDBP or DFHSPP during DWE processing for DFHTOR. It calls DFHTOR with end-LUW-cancel or end-LUW-commit code to perform cancel or commit of changes to TERMINAL, TYPETERM, CONNECTION, or SESSIONS definitions.

DFHTOR

Entry points: DFHTORNA

Called by: DFHAMP, DFHTON

Description: DFHTOR is the terminal object resolution program. DFHAMP calls DFHTOR for a TERMINAL, TYPETERM, CONNECTION, or SESSIONS object in a CICS system definition (CSD) file that is being installed, or when DFHAMP encounters an end-of-group. DFHTOR processes the objects and passes them to the terminal control builder program (DFHZCQ). The DFHTON entry is used for DWE processing.

DFHTORP

Entry points: DFHTORNA

Called by: DFHSII1

Description: DFHTORP is the terminal object recovery program. It is called during CICS initialization to purge TYPETERM and model terminal definitions from the catalog on a cold start, and to recover these definitions on an emergency restart.

DFHTPPA$, DFHTPP1$

Entry points: DFHTPPNA

Called by: DFHDSB, DFHM32

Description: The terminal page processor program handles DFHBMS TYPE=OUT, STORE, and RETURN requests. If OUT, DFHTPP sends the complete page using DFHTC macro requests; if STORE, the page is sent to temporary storage; and if RETURN, no output operation takes place but the page is returned to the application program.

The main subroutines of DFHTPP are:

```
TPNODDS - TYPE=STORE (PAGING) requests
TPOUT   - TYPE=OUT (TERMINAL) requests (the macro
          DFHTOM is used by both DFHTPP and DFHTPR
          to handle output to terminals)
TPRETPG - TYPE=RETURN (SET) requests.
```

Returns to: DFHPBP

DFHTPQ

Entry points: DFHTPQNA

Called by: DFHICP, DFHMCP, DFHTCP

Description: The undelivered messages cleanup program is initiated periodically in order to cancel the delivery of BMS messages that have been placed in temporary storage, but have remained undelivered for an interval exceeding the purge delay time interval specified by the PRGDLAY system initialization parameter, if this has a nonzero value.

DFHTPR

Entry points: DFHTPRNA

Called by: DFHMCP, DFHTCP

Description: The terminal page retrieval program (transaction CSPG) is invoked:

- By automatic transaction initiation as a result of a SCHEDULE issued by DFHTPS
- By a DFHPGLK LINK from DFHMCP, when CTRL=RETAIN or RELEASE on DFHBMS TYPE=PAGEOUT (RETAIN or RELEASE on SEND PAGE at command level)
- When CSPG or an operator paging command is entered at a terminal.

If the message is autopaged, DFHTPR retrieves the pages of the message in order, transmits them to the terminal, and then purges the message. Otherwise DFHTPR runs pseudo-conversationally. All further input is passed to DFHTPR, until the message is purged explicitly or implicitly. If the input is a valid paging command (page retrieval, page copy, page purge, or page chaining), it is processed. It is rejected if explicit purge is required, or passed back to normal task initiation if automatic purge is allowed.

The main subsections of DFHTPR are:

```
DFHMSPUT - Send error message to terminal
TPENCCHN - Encode and execute page chain
TPENCCOP - Encode and execute page copy
TPENCPUR - Execute page purge
TPENCRET - Encode page retrieval
TPERETA  - Reset to autopaging
TPERETQ  - Page query
TPEXIT   - Exit from program
TPEXPUR  - Execute page purge
TPEXRET  - Execute page retrieval
TPTSGET  - Get MCR or page from temporary storage.
```

DFHTPS

Entry points: DFHTPSNA

Called by: DFHICP, DFHMCP

Description: The terminal page scheduling program (transaction CSPS) is invoked for each terminal type to which a BMS logical message built with TYPE=STORE is to be sent. For each terminal designated by the originating application program, DFHTPR is scheduled to display the first page of the logical message if the terminal is in paging status, or the complete message if it is in autopage status. DFHTPS contains the following major subsections, each dealing with a separate function:

- DFHTPSNA—used when DFHTPS is invoked by automatic initiation on expiry of ICE, and as a result of an IC PUT request issued by DFHMCP (there is no associated terminal). This invocation schedules CSPG for terminals on this system, and schedules CSPS on the link to each remote system which owns terminals contained in the route list for the message (that is the function of TPS02000).
- TPS01000—used when DFHTPS is linked to from DFHMCP for direct paging requests to a terminal on a remote system. The task has a surrogate TCTTE as its primary facility, and owns a relay link connected to the terminal owning system. This section ships the pages of the message to the terminal-owning region, where it is re-created by the relay program (DFHAPRT) which issues BMS, STORE, TEXT, NOEDIT, and PAGEOUT requests.
- TPS02000—used when DFHTPS is scheduled by TPS01000 to run against the link to a remote system. This routine ships the logical message to the remote system and deletes the terminals on the remote system from the terminal list in the original message control record. (TPS03000 receives the information at the remote system.)
- TPS03000—used when DFHTPS is invoked by an ATTACH request from a remote system (that is, originated by TPS01000 or TPS02000). This routine receives the shipped logical message and issues BMS ROUTE, TEXTBLD, NOEDIT, and PAGEOUT requests to re-create the logical message on the terminal-owning region.

DFHTPS contains the following subroutine:

- TPSSHIPM—ships a complete logical message.

DFHTRAP

Entry points: DFHTRANA

Called by: DFHTRPT

Description: The FE global trap/trace exit is provided for diagnostic use only under the guidance of service personnel.

DFHTR640/AMDUSREF

Entry points: DFHTRPRG

Called by: IPCS

Description: The CICS GTF trace formatting routine is invoked by IPCS processing of the GTFTRACE keyword when a CICS entry (USR F6C, format ID X'EF') is encountered. For each entry, it writes a line containing the job name and then formats the entry in the same form as DFHTU640 does for an auxiliary trace print. AMDUSREF is defined as an alias for DFHTR640 because IPCS looks for a program called AMDUSRxx to format entries with format ID xx.

DFHTRP

Entry points: DFHTRPNA

Called by: Many AP domain modules

Description: The trace control program translates DFHTR, DFHTRACE, and DFHLFM macro requests to write trace entries into TR domain TRACE_PUT requests. DFHTRP collects the data required in the trace for the specified trace ID into a standard layout and issues the TRACE_PUT call. For requests to change the various trace flags that control tracing, DFHTRP issues KEDD format calls to the kernel domain.

DFHTRZCP

Entry points: DFHTRZCP

Called by: CEDA transaction, DFHTCRP, DFHTOR

Description: DFHTRZCP builds a terminal builder parameter set.

DFHTRZIP

Entry points: DFHTRZIP

Called by: CEDA transaction, DFHTCRP, DFHTOR

Description: DFHTRZIP builds a chain of builder parameter sets for sessions.

DFHTRZPP

Entry points: DFHTRZPP

Called by: CEDA transaction, DFHTCRP, DFHTOR

Description: DFHTRZPP builds a pool builder parameter set.

DFHTRZXP

Entry points: DFHTRZXP

Called by: CEDA transaction, DFHTCRP, DFHTOR

Description: DFHTRZXP builds a connection builder parameter set.

DFHTRZYP

Entry points: DFHTRZYP

Called by: CEDA transaction, DFHTCRP, DFHTOR

Description: DFHTRZYP builds a TYPETERM builder parameter set.

DFHTRZZP

Entry points: DFHTRZZP

Called by: CEDA transaction, DFHTCRP, DFHTOR

Description: DFHTRZZP merges a TYPETERM builder parameter set into a terminal builder parameter set.

DFHTSP

Entry points: DFHTSPNA

Called by: DFHACP, DFHAKP, DFHALP, DFHCRQ, DFHDBP, DFHDIP, DFHEDFP, DFHESE, DFHETS, DFHICP, DFHMCP, DFHMSP, DFHRTE, DFHSII1, DFHSTP, DFHTCBP, DFHTPP, DFHTPQ, DFHTPR, DFHTPS, DFHTSBP, DFHTSP, DFHTSRP, DFHZISP, DFHZRAQ, DFHZRAR, DFHZRSP

Description: The temporary-storage program services DFHTS requests. It maintains the tables, directories, and maps necessary to keep track of every temporary-storage record and of available space on the VSAM auxiliary storage or in main storage. The main subroutine of DFHTSP is DFHTSPAM, which manages auxiliary storage (including multiple buffers and strings).

DFHTU640

Entry points: DFHTRPRA

Called by: MVS

Description: The trace utility program formats and prints trace records stored on the auxiliary trace data set. This utility program is run as a separate job, and extracts selected trace entries as specified on parameter statements supplied as part of the input to the program.

DFHUCNV

Entry points: DFHUCNV

Called by: DFHCCNV

Description: DFHUCNV is a sample program for CICS OS/2 user data conversion. Users can write their own version of DFHUCNV to apply any conversion. If specified, a user-supplied conversion is applied before the standard conversion. DFHUCNV is invoked for each EXEC CICS request and reply that has resulted from a CICS OS/2 function shipping request and may require conversion of user data from ASCII to EBCDIC (inbound from CICS OS/2) or from EBCDIC to ASCII (outbound). DFHCCNV issues an EXEC CICS LINK to DFHUCNV before attempting any standard conversions. This allows a user program to convert data of type USERDATA, as defined in the CICS OS/2 conversion macros (DFHCNV).

The sample program obtains addressability to the COMMAREA passed to it, and checks that the request is a temporary-storage (TS) request. Then it checks that DFHCCNV managed to locate a conversion template for the resource (a TS queue) with this name, and scans and checks the template using the supplied template pointer and length. If the check is successful, the program translates the user data field as appropriate.

DFHUEH

Entry points: DFHUEHNA

Called by: CICS management modules containing exit points

Description: The user exit handler is the link between an exit point in a CICS management module in the AP domain, and the user code. DFHUEH invokes in turn each started exit program for that exit point, passing a parameter list defined in the CICS management module.

DFHUEM

Entry points: DFHUEMNA

Called by: DFHEIP

Description: The EXEC interface processor for the ENABLE, DISABLE, and EXTRACT user exit commands.

DFHUSBP

Entry points: DFHUSBNA

Called by: DFHRCRP

Description: The user backout program sends records, journaled by the user to the system log, to a user exit during emergency restart. The records are extracted by DFHRUP from the restart data set. They may exist for any logical unit of work, whether in flight or not, depending on the JCRSTRID value specified when the record was written.

DFHWCCS

Entry points: DFHWCCS

Called by: Many CAVM modules

Description: DFHWCCS provides common services for the CAVM:
- MVS FREEMAIN
- MVS GETMAIN
- MVS POST
- Message or MVS ABEND
- Create CAVM process block.

Returns to: MVS abend, caller

DFHWCGNT

Entry points: DFHWCGNA

Description: DFHWCGNT is the entry point list for CAVM modules above the 16MB line.

DFHWDATT

Entry points: DFHWDATT

Called by: DFHWDINA, DFHWMG1, DFHWMP1, DFHWSXPI

Description: DFHWDATT creates the CAVM process.

DFHWDINA

Entry points: DFHWDINA

Called by: DFHWSRTR

Description: DFHWDINA attaches the initial CAVM process. It sets up lock tables, the dispatcher control area, the LIFO control area, and the dispatcher ESPIE and ESTAE exits.

Returns to: DFHWDISP

DFHWDISP

Entry points: DFHWDISP, DFHWDIND

Called by: DFHWDWAT, DFHWDINA

Description: DFHWDISP is the CAVM process dispatcher. It dispatches the next ready CAVM process, or waits for an external event. It dispatches the initial CAVM process.

Returns to: Dispatched process, caller of DFHWDINA

DFHWDSRP

Entry points: DFHWDSRP

Called by: DFHWDINA, CAVM program check/abend

Description: DFHWDSRP establishes the ESPIE/ESTAE CAVM process. It performs CAVM process error handling for processes with ESPIE or ESTAE routines.

DFHWDWAT

Entry points: DFHWDWAT

Called by: Many CAVM modules

Description: DFHWDWAT causes the current CAVM process to wait for specific events.

Returns to: DFHWDISP

DFHWKP

Entry points: DFHWKPNA

Called by: DFHSTP

Description: DFHWKP takes a warm keypoint at the normal termination of CICS. This program is part of the restart component.

DFHWLFRE

Entry points: DFHWLFRE

Called by: Many CAVM modules

Description: DFHWLFRE frees the LIFO stack entry for CAVM modules running above the 16MB line.

DFHWLGET

Entry points: DFHWLGET

Called by: Many CAVM modules

Description: DFHWLGET gets the LIFO stack entry for CAVM modules running above the 16MB line.

DFHWMG1

Entry points: DFHWMG1

Called by: DFHWMI, DFHWDISP, DFHWDSRP

Description: DFHWMG1 is the main module of the CAVM message manager GET MESSAGE service. It is called by DFHWMI to initialize service, and attach itself as a message-reader CAVM process; by DFHWDISP to run as a message-reader CAVM process that reads messages and stores them; and by DFHWDSRP to handle ESPIE/ESTAE exits for the message reader.

DFHWMI

Entry points: DFHWMI

Called by: DFHWSXPI

Description: DFHWMI allocates the CAVM message-manager communication area. It calls each of the main message-manager modules, which then initialize themselves.

DFHWMMT

Entry points: DFHWMMT

Called by: DFHWMRD, DFHWMWR

Description: DFHWMMT provides VSAM GET and PUT services for the CAVM message data set.

DFHWMPG

Entry points: DFHWMPG

Called by: DFHWMP1, DFHWMWR

Description: DFHWMPG copies message data into the buffer provided by the user of PUTMSG, PUTREQ, PUTRSP, and CAVM message-manager services. It provides an ESPIE routine to handle program checks occurring during the copying.

DFHWMP1

Entry points: DFHWMP1

Called by: DFHWMI, DFHWDISP, DFHWDSRP

Description: DFHWMP1 is the main module of the CAVM message-manager PUT MESSAGE service. It is called by DFHWMI to initialize service, and attach itself as a message-writer CAVM process; by DFHWDISP to run as a message-writer CAVM process that writes messages to the CAVM message data set; and by DFHWDSRP to handle ESPIE and ESTAE exits for the message writer.

DFHWMQG

Entry points: DFHWMQG

Called by: DFHWMS20

Description: DFHWMQG runs under the CICS TCB above the 16MB line. It processes GETMSG CAVM message-manager requests. It waits for a message to arrive, then copies from the main-memory message queue created by the CAVM message-reader process.

DFHWMQH

Entry points: DFHWMQH

Called by: DFHWMG1, DFHWMQG

Description: The CAVM message-manager message input queue handler locates or creates message-queue anchor blocks, and adds copies of messages read by the CAVM reader process to the main-memory message queues.

DFHWMQP

Entry points: DFHWMQP

Called by: DFHWMS20

Description: DFHWMQP runs under the CICS TCB above the 16MB line. It processes CAVM message-manager PUTMSG, PUTREQ, and PUTRSP requests; places the request in the appropriate queue; and posts the queue to awaken CAVM process to handle request, waits for completion, and returns response to the caller.

DFHWMQS

Entry points: DFHWMQS

Called by: DFHWMP1, DFHWMWR

Description: The CAVM message-manager message output queue handler provides services to select the next work item to process, and posts items complete.

DFHWMRD

Entry points: DFHWMRD

Called by: DFHWMG1

Description: The CAVM message-manager message read routine reads messages from the CAVM message data set, taking account of the position of the active write cursor, and creates message blocks for copies of messages that have been read.

DFHWMS

Entry points: DFHWMSNA

Called by: Users of CAVM message services

Description: The CAVM message-manager service interface routine runs under the CICS TCB above the 16MB line. It builds a dummy CAVM process block, so that subsequent modules can run in an XRF LIFO environment, and calls DFHWMS20 to process a request passed by the caller.

DFHWMS20

Entry points: DFHWMS20

Description: The CAVM message manager services interface selects the request type and passes requests to DFHWMQP (PUTMSG, PUTREQ, PUTRSP) or DFHWMQG (GETMSG).

DFHWMWR

Entry points: DFHWMWR

Called by: DFHWMP1

Description: The CAVM message-manager message write routine takes data from PUTMSG requests and copies them into CI buffers to be written to the CAVM message data sets.

DFHWOS

Entry points: DFHWOSNA

Description: The overseer startup module loads DFHWOSA and passes control to it.

DFHWOSA

Entry points: DFHWOSNA

Called by: DFHWOS

Description: The overseer services initialization module processes control parameters, loads DFHWOSB, and sets up entry points for overseer services.

DFHWOSB

Entry points: DFHWOSNA

Called by: Overseer program

Description: The overseer service module processes requests from the overseer program which are issued by the DFHWOSM macro.

DFHWSRTR

Entry points: DFHWSMNA

Called by: DFHXRA, MVS after attach of new TCB

Description: The CAVM state-management request router and subtask entry point is the initial entry point for a CAVM task attached by DFHWSSN1 to process the CAVM SIGNON command. It calls DFHWSSN2 to continue the processing of the SIGNON request and, if it is accepted, calls DFHWDINA to attach the tick generator module DFHWSTI as the first and highest-priority CAVM process. It is called under the CICS TCB to queue the CAVM TAKEOVER command for processing by the CAVM task, and to initiate processing of the CAVM SIGNOFF command by detaching the CAVM task. DFHWSRTR is the initial entry point for MVS subtasks attached by the CAVM task to perform various functions, such as issuing requests for CSVC services, or formatting new CAVM data sets when they are used for the first time.

DFHWSSN1

Entry points: DFHWSSNA

Called by: DFHXRA

Description: DFHWSSN1 is the CAVM state management SIGNON initial entry point. The CICS task issues an MVS LINK, specifying load module DFHWSSON to perform a CAVM SIGNON request. DFHWSSN1 attaches the CAVM task to execute the request, waits to see if it is successful, detaches the task and, if it is not successful, reports the result to CICS.

DFHWSSN2

Entry points: DFHWSSN2

Called by: DFHWSRTR

Description: The CAVM state management SIGNON request handler is entered under the CAVM TCB to process a CAVM SIGNON request. It allocates storage for, and initializes, key CAVM control blocks, sets up DFHWSSOF as an ESTAE exit, calls DFHWSSN3 to OPEN the CAVM data sets, reads the state management record from the control data set, uses the JES inquire-job-status CSVC service provided by DFHWTI, and looks for surveillance signals from other CAVM users to check whether the environment is such that the requested SIGNON can be accepted. It prompts the operator for job status information if necessary. If SIGNON is accepted, it updates the state management record and status CIs to record that this job has signed on to the CAVM. When possible, it also cleans up out-of-date information in the CAVM data sets left behind by jobs that were unable to sign off properly before terminating.

DFHWSSN3

Entry points: DFHWSSN3

Called by: DFHWSSN2

Description: The CAVM state management data set initialization routine builds ACBs, and opens and validates the CAVM control and message data sets for CAVM SIGNON. It builds the reserve parameter list for serializing accesses to the control data set. If new CAVM data sets are being used for the first time, it attaches an MVS subtask to record relevant information in each data set's control interval, and to format the CIs needed by state management.

DFHWSSOF

Entry points: DFHWSSOF

Called by: MVS recovery/termination manager

Description: DFHWSSOF is the CAVM state management SIGNOFF request handler. During SIGNON processing, this module is established as an ESTAE exit for the CAVM task. It purges outstanding I/O requests, reads the state management record from the control data set, and searches it to see if this job has signed on to the CAVM. If so, it updates the status CI and state management record to indicate that the job has signed off. It makes the TAKEOVER message available to DFHWSRTR when an active system signs off after takeover has started.

DFHWSSR

Entry points: DFHWSSR

Called by: DFHWDISP

Description: The CAVM surveillance status reader runs as a process controlled by the XRF dispatcher, DFHWDISP. It reads the status CI of the partner system from the control data set or the message data set, generates internal CAVM events, and drives the NOTIFY exit when the partner's status changes, or its surveillance signals cease. For an alternate system, it monitors and records the time-of-day clock difference when the active system is running in a different CEC.

DFHWSSW

Entry points: DFHWSSW

Called by: DFHWDISP

Description: The CAVM surveillance status writer runs as a CAVM process controlled by the CAVM dispatcher, DFHWDISP. It writes a system's current status to its status CI in the control data set, or the message data set, to make it available to its partner and to provide a surveillance signal; generates an internal CAVM event when a status write completes; and puts the current time-of-day clock reading in the status CI to permit DFHWSSR to deduce the time-of-day clock difference when the active system and the alternate system are running in different CECs.

DFHWSTI

Entry points: DFHWSTI

Called by: DFHWDISP

Description: The CAVM surveillance tick generator and CICS status monitor runs as a CAVM process controlled by the CAVM dispatcher DFHWDISP. It issues an MVS STIMER for the surveillance interval and, when this expires, generates an internal CAVM clock-tick event, calls the inquire-CICS-status exit, and schedules the surveillance status writer processes, to cause a surveillance signal reporting this system's current status to be written to the control data set or the message data set.

DFHWSTKV

Entry points: DFHWSTKV

Called by: DFHWDISP

Description: The CAVM state management TAKEOVER request handler runs as a CAVM process controlled by the CAVM dispatcher DFHWDISP. When a new active SIGNON has been detected, it reads the state management record from the control data set and attaches an MVS subtask to invoke DFHWTI's validate-CLT CSVC service. When a TAKEOVER command has been issued, it reads the state management record, validates the TAKEOVER request, and attaches an MVS subtask to use DFHWTI's JES inquire-job-status service to determine the current state of the active system.

If the active system is still signed on to CAVM, it updates the state management record to indicate that a takeover is in progress, places the TAKEOVER message for the active system in the alternate system's status, and attaches an MVS subtask to invoke DFHWTI's TAKEOVER-initiate service.

After the active system has signed off (or terminated), it requests DFHWSSR to read the active system's final status, quiesces surveillance processing, and updates the state management record and status CIs to indicate the stage reached by takeover. It then arranges for surveillance processing to be resumed in active mode. It attaches an MVS subtask to invoke DFHWTI's process-CLT CSVC service if necessary.

When the active system has finally terminated, it updates the state management record to take its place as the new active system, generates internal CAVM events, and calls the NOTIFY exit to report the progress of the TAKEOVER request, including acceptability of the time-of-day clock reading. It terminates by returning to DFHWDISP.

DFHWSXPI

Entry points: DFHWSXPI

Called by: DFHWSTI

Description: The CAVM state management CAVM process initialization runs under the tick generator CAVM process towards the end of SIGNON. It attaches the TAKEOVER CAVM process (alternate systems only), two status writer CAVM processes, and two status reader CAVM processes, and then calls the CAVM message management initialization module.

DFHWTI

Entry points: DFHWTINA

Called by: DFHCSVC from: DFHWSSN2, DFHWSTKV, DFHZXSTS

Description: Takeover initiation is the primary function of this module, and is requested by CAVM state management at takeover to terminate the CICS active system issue commands in the CLT, and wait until the CICS active system terminates. Other XRF services provided by this module are to determine whether a job is running, to issue the operator commands for the overseer program, to issue MODIFY USERVAR to VTAM, to validate the CLT, and to process the CLT.

DFHWTRP

Entry points: DFHWTRP

Called by: Many CAVM modules

Description: DFHWTRP makes a trace entry in the CAVM main-memory trace table.

DFHXCP

Entry points: DFHXCPNA

Called by: DFHKCP

Description: DFHXCP processes DFHKC CANCEL, CHAP, RESUME, SUSPEND, and WAIT macro calls to the transaction manager.

DFHXCPC

Entry points: DFHXCPC

Called by: DFHKCP

Description: DFHXCPC processes DFHKC ATTACH, CHANGE, DEQ, DEQALL, ENQ, and SRB macro calls to the transaction manager. It receives DFHKC INITIALIZE, REPLACE, and WAITINIT macro calls to the transaction manager and passes them on to DFHKCQ.

DFHXCP1

Entry points: DFHXCP1

Called by: DFHXCPC

Description: DFHXCP1 finds a new range of free transaction numbers when the current range has been used up.

DFHXFP

Entry points: DFHXFPNA

Called by: DFHISP, DFHMIRS

Description: The online data transformation program takes data addressed from a parameter list (command-level or DL/I), and constructs an FMH suitable for transmission to a remote ISC or MRO system; DFHXFP also performs the reverse transformation.

DFHXFQ

Entry points: DFHXFQNA

Called by: DFHXEPRH

Description: The batch data transformation program executes in an EXCI region. DFHXFQ takes data addressed from a DPL parameter list and constructs an FMH suitable for passing to the online region; DFHXFQ also performs the reverse transformation.

DFHXFX

Entry points: DFHXFXNA

Called by: DFHISP, DFHMIRS

Description: DFHXFX performs the same logical transformations of function shipping requests as DFHXFP but in a manner that is optimized for the MRO environment. It is not used for the transformation of DL/I requests; these are processed by DFHXFP.

DFHXMP

Entry points: DFHXMPNA

Called by: DFHDRPE, DFHZCX

Description: The cross-memory program is invoked by a program call (PC) instruction and uses MVS cross-memory services to pass data from one subsystem to another within the same processing unit. The communicating subsystems are usually in CICS address spaces, but DFHXMP does not assume this.

DFHXRA

Entry points: DFHXRANA

Called by: DFHAPDM, DFHCSSC, DFHCXCU, DFHDBCR, DFHDBCT, DFHSIC1, DFHSII1, DFHSTP, DFHTCRP, DFHTDRP, DFHXRCP, DFHXRSP, DFHZNAC, DFHZOPN, DFHZSLS

Description: DFHXRA is the program that executes the DFHXR macro. It runs under the CICS TCB in AMODE(24). In general, it uses CICS macros to invoke other services. Exceptions are MVS LINK to DFHWSSON to sign on to the CAVM, and MVS LOAD and DELETE for DFHWSMS to sign off from the CAVM, and to initiate takeover. It invokes global user exit XXRSTAT, which can lead to the abend 208.

DFHXRB

Entry points: DFHXRBNA

Called by: DFHWDSRP, DFHWMQH, DFHWMRD, DFHWSSR, DFHWSTKV

Description: DFHXRB is the XRF notify exit program. Its address is passed to the CAVM when CICS signs on to the CAVM. It runs under the CAVM TCB in AMODE(31); reacts to events detected by various CAVM modules; and creates a queue of work elements (chained from XRWECHN) to be processed by DFHXRSP.

DFHXRC

Entry points: DFHXRCNA

Called by: DFHWSSN2, DFHWSTI

Description: DFHXRC is the CICS-status exit program. Its address is passed to the CAVM when CICS signs on to the CAVM. It runs under the CAVM TCB in AMODE(31), and returns the latest CICS-status data to be written to the state management data set.

DFHXRCP

Entry points: DFHXRCNA

Description: The XRF console communication task runs under the CICS TCB in AMODE(24). It processes MODIFY commands received by CICS during initialization of the alternate system. It initiates takeover, shuts down the active system, and manages trace and dump as required.

DFHXRE

Entry points: DFHXRENA

Called by: DFHPCP

Description: The XRF startup program is the entry point for the system task attached by DFHXRA. It links to DFHXRE, whichever module was indicated by DFHXRA.

DFHXRP

Entry points: DFHXRANA

Called by: Not applicable

Description: DFHXRP consists of six object modules link-edited together:

```
DFHXRA - XRF request processor
DFHXRB - XRF NOTIFY exit program
DFHXRC - XRF inquire status exit program
DFHXRE - XRF startup program
DFHXRF - XRF CAVM sign-off interface
DFHWMS - CAVM message manager service interface.
```

It is loaded by DFHSIB1.

DFHXRSP

Entry points: DFHXRSNA

Called by: DFHXRA

Description: DFHXRSP is the XRF surveillance program, which runs as a program under a CICS transaction. It runs under the CICS TCB in AMODE(31); processes the queue of work elements created by DFHXRB; attaches the catch-up transaction CXCU, initiates takeover, and shuts down CICS as required; and can issue abends 206 and 207.

DFHXSMN

Entry points: DFHXSMNA

Called by: DFHBSTS, DFHCRNP, DFHDLIDP, DFHDLIRP, DFHEDFP, DFHEIPSE, DFHSII1, DFHSUSN, DFHSUXS, DFHTACP, DFHZSUP

Description: The security manager is invoked by the DFHSEC macro, and provides an interface to the external security manager (ESM). DFHXSMN validates the parameters passed, then calls DFHXSMX as a general-purpose subroutine to invoke the ESM.

DFHXSMX

Entry points: DFHXSMNA

Called by: DFHXSMN

Description: DFHXSMX is the subroutine used by the security manager to invoke the external security manager (ESM). For resource checking, this routine first issues the MVS RACROUTE REQUEST=FASTAUTH macro, which calls the ESM in problem state. All other security functions require the caller to be in supervisor state. For these functions, and for a failed FASTAUTH call that requires logging, the CICS SVC is issued under a general purpose subtask, entered by the DFHSK macro, to shield the main CICS task from any imbedded waits that may occur in the ESM.

DFHXSS

Entry points: DFHXSSNA

Called by: DFHCSVC

Description: DFHXSS invokes the external security manager (ESM) for all functions that need to be invoked while authorized, except for the EXTRACT functions for which it passes control to DFHXSSB.

DFHXSSB

Entry points: DFHXSSB

Called by: DFHXSS

Description: This module extracts data from the ESM's database. DFHXSSB extracts userid-related data at signon time, and session key information at LU6.2 session bind time. It uses the MVS RACROUTE REQUEST=EXTRACT macro.

DFHXSWM

Entry points: DFHXSWM

Called by: DFHXSMN

Description: DFHXSWM passes and retrieves messages to and from the XRF alternate system to see if security initialization is required in the XRF environment.

DFHXTCI

Entry points: DFHXTCI

Description: DFHXTCI is the transaction invoked when the alternate system begins a takeover. It examines the TCT to locate the terminals with XRF backup sessions, and queues these TCTTEs to DFHZSES for the SESSIONC CONTROL=SWITCH command.

DFHXTP

Entry points: DFHXTPNA

Called by: DFHTPS, DFHZTSP, DFHZXRL, DFHZXRT

Description: The terminal sharing transformation program comprises four logical modules (known as transformers 1 through 4). DFHXTP transforms routing requests into the LU type 6 format for shipping to a remote CICS address space.

DFHZABD

Entry points: DFHZABD1

Called by: TC CTYPE= requests

Description: If a TC CTYPE request is issued when ZCP has been generated without VTAM support, DFHZABD is invoked to abend the transaction.

DFHZACT

Entry points: DFHZACT1

Called by: DFHZDSP

Description: The activate scan routine scans the four TCTTE activity queues: activate, log, wait, and NACP. DFHZACT scans the activate queue for request bits that may be set in the TCTTEs; for each request, DFHZACT calls the appropriate module. If no requests are outstanding, the TCTTE is removed from the queue. If the NACP queue is not empty, DFHZACT attaches DFHZNAC (if not already attached). Similarly, if the log queue is not empty, DFHZACT attaches DFHZRLG. DFHZACT scans the wait queue. If automatic resource definition is in the system, DFHZACT looks for any corresponding work elements. For each work element, DFHZATA is attached.

DFHZAIT

Entry points: DFHZAIT1

Called by: DFHSIF1

Description: The attach initialization tables routine initializes local tables used by the mainline task-attach routine, DFHZATT. DFHZAIT generates the page command table from information supplied by the system initialization table, modifying it for use by DFHZATT. DFHZAIT also initializes the transaction code delimiter table.

DFHZAND

Entry points: DFHZAND1

Called by: DFHZARQ

Description: The abend control block builder is used to assist in building the transaction abend block when an abend has occurred in an interconnected system. Its function is to extract the error sense bytes, and the diagnostic message sent by the other system, and to copy these into the block. As an initial step in its processing, DFHZAND acquires storage for the block itself.

DFHZARER

Entry points: DFHZARER

Called by: DFHZARL, DFHZARR, DFHZARRA

Description: DFHZARER tidies up after an LU6.2 protocol error or session failure has been detected. For some errors, it calls DFHZNAC.

DFHZARL

Entry points: DFHZARL1

Called by: DFHACP, DFHCPCBA, DFHCPCLC, DFHCRS, DFHEGL, DFHETL, DFHLUP, DFHXFP, DFHXTP, DFHZARL, DFHZARM, DFHZERH, DFHZISP, DFHZLUS, DFHZSUP, DFHZTSP, DFHZXRL, DFHZXRT

Description: DFHZARL is called via the DFHLUC macro, which passes the LU6.2 request in a parameter list mapped by the DFHLUCDS DSECT. If the request is for a remote APPC device, DFHZARL passes the parameter list to DFHZXRL for processing. (APPC is advanced program-to-program communication.) Otherwise, it examines the parameter list to determine the function required. Most functions are processed by DFHZARL. However, it calls the following modules as indicated:

```
DFHZARER - Protocol errors and exceptions
DFHZARR  - RECEIVE requests
DFHZARRA - FREE-STORE requests
DFHZERH  - Handling FMH7s and negative responses
DFHZISP  - ALLOCATE and FREE requests
DFHZRVL  - Receiving SNA indicators from VTAM
DFHZSDL  - Sending data to VTAM.
```

It also manages the logical receive buffer pointers TCTERBLA and TCTERBLL in a consistent manner with the physical receive buffer pointers TCTERBA and TCTERBDL, as (address, length) pairs.

DFHZARM

Entry points: DFHZARM1

Called by: DFHZARQ, DFHETL, DFHZISP

Description: DFHZARM handles DFHTC macros for LU6.2 sessions.

DFHZARQ

Entry points: DFHZARQ1

Called by: DFHETC, DFHTC macro

Description: The application request interface module analyzes the terminal control request from the application. For a VTAM terminal, it sets the appropriate flags and calls the required module or adds the TCTTE to the activate chain.

DFHZARR

Entry points: DFHZARR

Called by: DFHZARL

Description: DFHZARR controls the receive function for LU6.2 application requests. It calls DFHZARRC to decide what to process next, or whether it is necessary to call its inline subroutine DFHZARR1 to receive more data. Then it processes the returned item, and decides whether the receive is complete. If the receive is not complete, DFHZARR loops, calling DFHZARRC and processing the returned item, until enough data has been received. DFHZARR uses the inline subroutine DFHZARR0 and the DFHZARRA module to control various receive buffers. It also uses DFHZARRF to receive FMH7s and negative responses, DFHZUSR to control the conversation state, and the inline subroutine DFHZARR1 to handle the type of receive and how much data is to be received.

DFHZARR0 is responsible for updating the logical buffer pointers TCTERBLA and TCTERBLL, shifting up data in the LU6.2 receive buffer, and resetting associated indicators, for example, TCTECCDR in the TCTTE LUC extension.

DFHZARR1 is responsible for setting fields TCTEMINL and TCTEMAXL in the TCTTE LUC extension to inform DFHZRVL how much data to receive and whether the request is a receive immediate or a receive and wait. DFHZARR1 calls DFHZARR0 to shift up data in the LU6.2 receive buffer, and then calls DFHZRVL to receive RUs from VTAM by placing requests on the active chain.

DFHZARRA

Entry points: DFHZARRA

Called by: DFHZARL, DFHZARR

Description: DFHZARRA controls all functions concerned with the LU6.2 application receive buffer. These include GETMAIN and FREEMAIN of buffers, copying data into a buffer, and updating the pointer to the next free slot.

DFHZARRC

Entry points: DFHZARRC

Called by: DFHZARR

Description: DFHZARRC is responsible for examining what has been received from VTAM on a particular session (for example, data, PS headers, FMH7s, and indicators), and for deciding what should be processed next on behalf of the application. The result is returned to DFHZARR.

DFHZARRF

Entry points: DFHZARRF

Called by: DFHZARR

Description: DFHZARRF receives LU6.2 FMH7s and negative responses. It calls the DFHZARR0 subroutine to shift up data in the LU6.2 receive buffer, and then calls DFHZERH.

DFHZASX

Entry points: DFHZASX1

Called by: VTAM

Description: The asynchronous command exit module is called by VTAM if an asynchronous command is received. The only such commands are request shutdown, quiesce at end of chain, release quiesce, and signal. DFHZASX sets up the TCTTE appropriately and returns control to VTAM.

DFHZATA

Entry points: DFHZATA

Called by: DFHZACT

Description: The autoinstall program runs as the CATA transaction and performs operations necessary to INSTALL autoinstallable terminals. It requests information from a user program where appropriate.

DFHZATD

Entry points: DFHZATD

Called by: DFHZACT, DFHZNAC

Description: The autoinstall delete program runs as the CATD transaction and performs operations necessary to DELETE autoinstalled terminals. It requests information from a user program where appropriate.

DFHZATDX

Entry points: DFHZATDX

Called by: DFHZATA, DFHZATD

Description: DFHZATDX is the user program for autoinstall. It is called when:

- An autoinstall INSTALL is in progress
- An autoinstall DELETE has just completed
- An autoinstall INSTALL has failed.

For INSTALL, DFHZATDX selects a model name and the corresponding TRMIDNT to be used by the terminal control builder program (DFHTBSxx). This program can be used as a model for a user program.

DFHZATI

Entry points: DFHZATI1

Called by: DFHZACT

Description: The automatic task initiation module checks for stress conditions, calls DFHZSIM if the node is not in session, acquires an RPL if necessary, and issues a conditional DFHKC TYPE=AVAIL macro. DFHZATI initiates bid protocols to decide whether the LU is available.

DFHZATMD

Entry points: DFHZATMD

Called by: DFHZATMF

Description: This program deletes all remote terminal definitions that are flagged (by DFHZATMF) for deletion.

DFHZATMF

Entry points: DFHZATMF

Called by:

Description: This program flags remote terminals for Mass-deletion (by DFHZATMD). It is a part of the transaction routing component, and is started to flag all skeletons that have been unused for more than the terminal latency period for deletion.

DFHZATR

Entry points: DFHZATR

Called by: DFHZATR, DFHZXRE0

Description: The autoinstall restart program runs as the CATR transaction at CICS startup after the time period specified in the AIRDELAY parameter. DFHZATR scans all autoinstalled terminals, and causes the CATD transaction to be called to delete any autoinstalled terminals that have not been used during the AIRDELAY interval.

DFHZATS

Entry points: DFHZATS

Called by: DFHZTSP, DFHCRS

Description: The remote autoinstall program runs as the following four transactions:

CITS The remote autoinstall function that is attached by DFHZTSP.

CDTS The remote delete function that is attached by DFHZTSP or DFHCRS.

CFTS The remote reset function that flags terminals for mass deletion after a CICS restart and is attached by DFHZTSP or DFHCRS.

CMTS The mass delete function of remote terminals that is attached by DFHZATS transaction CFTS if it finds any terminals for deletion.

DFHZATT

Entry points: DFHZATT1

Called by: DFHZACT

Description: The task attach module checks for stress conditions, allocates an RPL if necessary, and determines the task to be attached either from the data, or from the TCTTE (if the previous transaction specified TRANID), or from the AID (for a 3270). DFHZATT also checks for paging commands (having been modified by DFHZAIT). Finally a conditional ATTACH is issued. The module is applicable for VTAM, SRL, and MVS console support.

DFHZBAN

Entry points: DFHZBAN

Called by: DFHZOPN

Description: The terminal control bind analysis program checks that a bind is valid and supportable and, if requested, sets the TCTTE information that supports the session parameters.

DFHZBKT

Entry points: DFHZBKT1

Called by: DFHZSDL, DFHZSLX, DFHZRLX, DFHZLUS

Description: DFHZBKT maintains the bracket state for LU6.2.

DFHZBLX

Entry points: DFHZBLX

Called by: DFHZSCX

Description: DFHZBLX is the part of of SCIP exit which processes LU6.2 binds. It matches a TCTTE to the BIND and schedules DFHZOPN to complete the BIND process. This module returns to VTAM.

DFHZCA

Entry points: DFHZCANA

Called by: See component submodules

Description: DFHZCA is the name of the load module created when the following modules are link-edited together:

```
DFHZACT  - Activate scan
DFHZFRE  - FREEMAIN request
DFHZGET  - GETMAIN request
DFHZQUE  - Chaining
DFHZRST  - RESETSR.
```

DFHZCB

Entry points: DFHZCBNA

Called by: See component submodules

Description: DFHZCB is the name of the load module created when the following modules are link-edited together:

DFHZATI
: Automatic task initiation

DFHZDET
: Task detach

DFHZHPSR
: HPO send/receive

DFHZLRP
: Logical record presentation

DFHZRAC
: Receive-any completion

DFHZRAS
: Receive-any slowdown processing

DFHZRVS
: Receive specific

DFHZRVX
: Receive specific exit

DFHZSDR
: Send response

DFHZSDS
: Send DFSYN

DFHZSDX
: Send DFSYN data exit

DFHZSSX
: Send DFSYN exit

DFHZUIX
: User input exit

DFHZCC

Entry points: DFHZCCNA

Called by: See component submodules

Description: DFHZCC is the name of the load module created when the following modules are link-edited together:

DFHZARER
: LU6.2 protocol error and exception handler

DFHZARL
: LU6.2 application request logic

DFHZARM
: LU6.2 migration logic

DFHZARR
: LU6.2 application receive request logic

DFHZARRA
: LU6.2 application receive buffer support

DFHZARRC
: LU6.2 classify what next to receive

DFHZARRF
LU6.2 receive FMH7 and ER1

DFHZBKT
LU6.2 bracket state machine

DFHZCHS
LU6.2 chain state machine

DFHZCNT
LU6.2 contention state machine

DFHZCRT
LU6.2 RPL_B state machine

DFHZRLP
LU6.2 post-VTAM receive logic

DFHZRLX
LU6.2 receive exit program

DFHZRVL
LU6.2 pre-VTAM receive logic

DFHZSDL
LU6.2 send logic

DFHZSLX
LU6.2 send exit program

DFHZSTAP
MRO or LU6.2 conversation state determination

DFHZUSR
LU6.2 conversation state machine

DFHZCHS

Entry points: DFHZCHS1

Called by: DFHZRLX, DFHZSDL, DFHZSLX

Description: DFHZCHS maintains the chain state for LU6.2.

DFHZCLS

Entry points: DFHZCLS1

Called by: DFHZACT

Description: The close destination module obtains an RPL if necessary, issues CLSDST to VTAM, and checks if it was accepted. The CLSDST exit handles the completion of the request. DFHZCLS performs a normal closedown procedure according to the LU type (for example, LU6 sends SBI and BIS). In the case of an abnormal closedown, DFHZCLS performs immediate termination, using CLSDST or TERMSESS commands. If the terminal was automatically defined, it is put out of service.

DFHZCLX

Entry points: DFHZCLX1

Called by: VTAM

Description: The close destination exit module receives control from VTAM when a CLSDST or TERMSESS request completes. If the CLSDST or TERMSESS was successful, DFHZCLX cleans up TCTTE and returns to VTAM; otherwise it enqueues the TCTTE to DFHZNAC and then returns to VTAM.

DFHZCNA

Entry points: DFHZCNA1

Called by: DFHZDSP

Description: The system console activity control program is responsible for CICS system requests. It performs the following functions:

- Shutdown—when all other access method terminals have been quiesced, quiesces console support, allowing CICS to terminate.
- Resume—resumes tasks waiting on read request when they are completed.
- Detach—releases all TIOAs associated with a completed task.
- Attach—passes the data associated with a MODIFY command (in a TIOA attached to a console TCTTE) to DFHZATT to create a task.
- ATI—determines whether a console TCTTE is available for automatic task initiation.

DFHZCNR

Entry points: DFHZCNR1

Called by: DFHZARQ

Description: The system console application request program performs READ, WRITE, and CONVERSE operations to an MVS system console that is used as a terminal.

DFHZCNT

Entry points: DFHZCNT1

Called by: DFHZLUS, DFHZRLX

Description: DFHZCNT maintains the contention state for LU6.2.

DFHZCP

Entry points: DFHZCPNA

Called by: See component submodules

Description: DFHZCP is the name of the load module created when the following modules are link-edited together:

```
DFHZARQ  - Application request handler
DFHZATT  - Attach routine
DFHZCNA  - System console activity control
DFHZDSP  - Dispatcher
DFHZISP  - Allocate/free/point routine
DFHZSUP  - Startup task
DFHZUCT  - 3270 uppercase translation.
```

DFHZCQ

Entry points: DFHZCQ

Called by: DFHAMTP, DFHCRS, DFHQRY, DFHTCRP, DFHWKP, DFHZATA, DFHZATD, DFHZTSP, DFHZXCU

Description: DFHZCQ is the control program for all requests for the dynamic add and delete of terminal control table entries. It is called by resource definition online (RDO) to:
- Cold start group lists
- Cold or warm start nonmigrated VTAM resources
- Dynamically install using the CEDA transaction.

The main subroutines of DFHZCQ are:

```
DFHZCQCH - Catalog a TCT element
DFHZCQDL - Delete
DFHZCQIN - Initialize DFHZCQ
DFHZCQIQ - Inquire about TCTTE
DFHZCQIS - Install TCTTE
DFHZCQIT - Add macro-generated TCTTE
DFHZCQRS - Restore ZC resource.
```

DFHZCQDL

Entry points: DFHZCQDL

Called by: DFHZCQ00, DFHZNAC, RDO

Description: DFHZCQDL dynamically deletes a TCT entry when the entry is quiesced. This module is part of DFHZCQ.

DFHZCQIN

Entry points: DFHZCQIN

Called by: DFHTCRP

Description: DFHZCQIN initializes DFHZCQ for all its operations. This module is part of DFHZCQ.

DFHZCQIQ

Entry points: DFHZCQIQ

Called by: DFHZTSP

Description: DFHZCQIQ obtains the parameters for a TCT resource and is called by DFHZTSP in the terminal-owning node as part of the process of shipping a TCT definition to a remote system. This module is part of DFHZCQ.

DFHZCQIS

Entry points: DFHZCQIS

Description: DFHZCQIS installs a TCTTE. If the resource already exists, the old resource is deleted.

DFHZCQIT

Entry points: DFHZCQIT

Description: DFHZCQIT adds a macro-generated TCTTE to a CICS system.

DFHZCQRS

Entry points: DFHZCQRS

Description: During emergency restart or warm start, DFHTCRP restores terminal control resources to the state they were in before the last shutdown of CICS, using the restart data set.

DFHZCRQ

Entry points: DFHZCRQ1

Called by: TC CTYPE requests

Description: The CTYPE request module analyzes DFHTC CTYPE commands, and calls or links to the appropriate send module.

DFHZCRT

Entry points: DFHZCRT1

Called by: DFHZACT, DFHZARL, DFHZFRE, DFHZNAC, DFHZRAC, DFHZRLP, DFHZRVL, DFHZSDL, DFHZSHU, DFHZSTU, DFHZTPX

Description: DFHZCRT maintains the RPL_B state for LU6.2.

DFHZCUT

Entry points: DFHZCUT

Called by: DFHCSSC, DFHLUP, DFHSNAT, DFHTCPLR

Description: DFHZCUT manages the persistent verification signed-on-from list, also known as the local userid table (LUIT). There is one LUIT per connection supporting persistent verification.

DFHZCW

Entry points: DFHZCWNA

Called by: See component submodules

Description: DFHZCW is the name of the load module created when the following modules are link-edited together:

```
DFHZERH  - LU6.2 error program
DFHZEV1  - LU6.2 BIND security
DFHZEV2  - LU6.2 BIND security
DFHZLUS  - LU6.2 session management program.
```

DFHZCX

Entry points: DFHZCXNA

Called by: See component submodules

Description: DFHZCX is the name of the load module created when the following modules are link-edited together:

```
DFHZABD  - Abend routine for incorrect requests
DFHZAND  - Build TACB before issuing PC abends
DFHZCNR  - System console application request
DFHZIS1  - ISC or IRC syncpoint
DFHZIS2  - IRC internal requests
DFHZLOC  - Locate TCTTE and ATI requests
DFHZSTU  - Terminal control status change.
```

DFHZCXR

Entry points: DFHZCXRA

Called by: See component submodules

Description: DFHZCXR is the generic name allocated to a composite module that is not called by any other code. It includes the following transaction-routing related modules:

```
DFHZTSP  - Terminal-sharing program
DFHZXRL  - Routes LU6.2 commands to TOR
DFHZXRT  - Receives LU6.2 commands from AOR.
```

DFHZCY

Entry points: DFHZCYNA

Called by: See component submodules

Description: DFHZCY is the name of the load module created when the following modules are link-edited together:

DFHZASX
: DFASY exit

DFHZDST
: SNA-ASCII translation

DFHZLEX
: LERAD exit

DFHZLGX
: LOGON exit

DFHZLTX
: LOSTERM exit

DFHZNSP
: Network services exit

DFHZOPA
: Open VTAM ACB

DFHZRRX
: Release request exit

DFHZRSY
: Resynchronization

DFHZSAX
: Send synchronous command exit

DFHZSCX
: SESSION control input exit

DFHZSDA
: Send synchronous command

DFHZSES
: SESSIONC

DFHZSEX
: SESSIONC exit

DFHZSHU
: Shutdown VTAM

DFHZSIM
: SIMLOGON

DFHZSIX
: SIMLOGON exit

DFHZSKR
: Send response to command

DFHZSLS
: Set logon start

DFHZSYN
: Handle CTYPE=SYNC or CTYPE=RECOVER request

DFHZSYX
: SYNAD exit

DFHZTPX
: TPEND exit

DFHZTRA
: Create ZCP or VIO trace requests

DFHZXRC
: XRF session state data analysis

DFHZCZ

Entry points: DFHZCZNA

Called by: See component submodules

Description: DFHZCZ is the name of the load module created when the following modules are link-edited together:

```
DFHZCLS  - CLSDST
DFHZCLX  - CLSDST exit
DFHZCRQ  - Command request
DFHZEMW  - Error message writer
DFHZOPN  - OPNDST
DFHZOPX  - OPNDST exit
DFHZRAQ  - Read-ahead queuing
DFHZRAR  - Read-ahead retrieval
DFHZTAX  - Turnaround exit.
```

DFHZDET

Entry points: DFHZDET1

Called by: DFHZACT, DFHZISP

Description: The task detach module receives control when a detach request is issued by DFHZISP. If a WRITE is pending (deferred write or any write), the SEND routine is called. If the SEND cannot complete, the DETACH request is left on the activate queue. If requests are queued then DFHZACT drives DFHZDET when the operation is complete. If the node is in between bracket state, an end bracket is sent.

DFHZDSP

Entry points: DFHZDSP1

Called by: DFHSII1

Description: The dispatcher module handles the dispatching of modules for execution, and gives control to VTAM modules of ZCP using DFHZACT.

DFHZDST

Entry points: DFHZDST1

Called by: DFHZRVX, DFHZSDS

Description: The data stream translator module translates data between EBCDIC and ASCII code while that data is being sent and received on VTAM sessions.

DFHZEMW

Entry points: DFHZEMW1

Called by: DFHACP, DFHZDET, DFHZNAC, DFHZRAC

Description: The error message writer module handles all requests for error messages on VTAM supported terminals/LUs. According to the request flags, it:
- Sends a negative response
- Purges unprocessed inbound data until EOC or CANCEL is received
- Sends an error message.

DFHZERH

Entry points: DFHZERH1

Called by: DFHZARL, DFHZARRF

Description: DFHZERH handles the sending and receiving of LU6.2 FMH7s and negative responses. It also manages the logical receive buffer pointers TCTERBLA and TCTERBLL in a consistent manner with the physical receive buffer pointers TCTERBA and TCTERBDL, as (address, length) pairs.

DFHZEV1

Entry points: DFHZEV11

Description: DFHZEV1 is the LU6.2 bind-time security encryption validation program, part 1.

DFHZEV2

Entry points: DFHZEV21

Description: DFHZEV2 is the LU6.2 bind-time security encryption validation program, part 2.

DFHZFRE

Entry points: DFHZFRE1

Called by: DFHZACT, DFHZEMW, DFHZCLS, DFHZCLX

Description: The FREEMAIN module is used to free storage (RPLs, NIBs, bind areas, TIOAs, buffer lists, LUC send/receive buffers, and extract logon data) acquired by ZC modules. Some storage is also freed by other ZC modules.

DFHZGET

Entry points: DFHZGET1

Called by: DFHZACT, DFHZARL, DFHZATI, DFHZATT, DFHZCLS, DFHZISP, DFHZOPN, DFHZRAC, DFHZRST, DFHZRSY, DFHZRVL, DFHZRVS, DFHZSDA, DFHZSDL, DFHZSDR, DFHZSDS, DFHZSES, DFHZSKR

Description: The GETMAIN module is used to acquire an RPL, NIB, bind area, TIOA, buffer list, or LUC send/receive buffer. DFHZGET also sets up the dynamic NIB using the information in the NIB descriptor block. Normally, when a ZC module requires some of the above storage, it invokes DFHZGET to obtain the storage; if this is unsuccessful, it may queue the request, and then DFHZACT calls DFHZGET on behalf of the caller.

DFHZHPRX

Entry points: DFHZHPNA

Called by: DFHKCSP (via DFHZHPSR and DFHKCP)

Description: In authorized path SRB mode, DFHZHPRX issues VTAM EXECRPL.

DFHZHPSR

Entry points: DFHZHPS1

Called by: DFHZRVS, DFHZSDS

Description: DFHZHPSR is the SEND and RECEIVE module for the HPO environment.

DFHZISP

Entry points: DFHZISP1

Called by: DFHISP, DFHKCP

Description: The intersystem program services ISC requests to free, or point to, a particular TCTTE within a specified system, or to allocate a TCTTE within a specified system. DFHZISP also handles ATI requests, and checks for a terminal time-out.

DFHZIS1

Entry points: DFHZIS11

Description: DFHZIS1 handles the transmissions control CTYPE requests of Prepare, Syncpoint Request (SPR), Commit, and Abort. Each request is translated into the appropriate ISC/IRC action and is transmitted to the connected system.

DFHZIS2

Entry points: DFHZIS21

Called by: DFHZARQ, DFHZIS1

Description: The intersystem program provides services for CICS system code that wants to use intersystem or interregion (IRC) function requests:

RECEIVE
: Is invoked when DFHCRNP gets input data as a result of a ‘switch first’ SVC request.

IOR
: The IRC input/output routine. This interfaces with the IRC SVC in order to send data to the other end of the connection, or await data from there.

GETDATA
: Is used to fetch input data into a TIOA.

DISCONNECT
: Disconnects a given IRC link.

STOP
: Quiesces interregion activity, either for connections to a given system, or for the whole of IRC.

LOGOFF
: Issues a logoff request to the IRC SVC. This completes IRC activity for this CICS system.

OPERATIVE
: Allows connections to be made to a given system.

RECABRT
: processes input abend FMHs (FMH07).

DFHZLEX

Entry points: DFHZLEX1

Called by: VTAM

Description: The logical error address (LERAD) exit module receives control from VTAM when a logical error is detected. Logical errors are usually the result of an incorrectly defined terminal table.

DFHZLGX

Entry points: DFHZLGX1

Called by: VTAM

Description: The logon exit module receives control from VTAM when a terminal logs on to the network. DFHZLGX scans the CICS NIBs and, if a match is found, sets an OPNDST request in the corresponding TCTTE and places it on the activate queue. If no match is found, DFHZLGX defines a terminal automatically, if possible, by allocating an autodefine work element which holds the CINIT_RU. The work element is then queued for activate scan processing. Otherwise, a dummy TCTTE is placed on the NACP queue to write an error message.

DFHZLOC

Entry points: DFHZLOC1

Called by: DFHTC CTYPE=LOCATE

Description: The locate module provides two functions:
- Locates specific TCTTEs, TCTSEs, and SESSIONs in the TCT
- Locates LDC information.

DFHZLRP

Entry points: DFHZLRP1

Called by: DFHZARQ, DFHZSUP

Description: The logical record presentation module handles deblocking of input data. The delimiters that are recognized are new line (NL), interchange record separator (IRS), and transparent (TRN). One logical record is returned for each DFHTC TYPE=READ request.

DFHZLTX

Entry points: DFHZLTX1

Called by: VTAM

Description: The lost terminal (LOSTERM) exit module receives control when VTAM detects a loss of contact with a node. There are three possible return codes set by VTAM on entry to this routine:

node lost, recovery in progress
: The terminal is placed out of service with no further action taken.

node lost, recovery successful
: The TCTTE is queued to the NACP queue with a ‘successful’ error code set; NACP issues a CLSDST, schedules a SIMLOGON, and issues an information message.

node lost, no recovery or unsuccessful recovery
: The TCTTE is queued to the NACP queue with an ‘unsuccessful’ error code set; NACP issues a

CLSDST and also the appropriate message.

DFHZLUS

Entry points: DFHZLUS1

Description: DFHZLUS handles session management for LU6.2 sessions.

DFHZNAC

Entry points: DFHZNANA

Called by: DFHZACT

Description: The node abnormal condition program is attached by DFHZACT when an error in communication with a logical unit occurs. DFHZNAC performs the following functions:
- Analyzes abnormal conditions
- Sends appropriate messages to the CSNE transient data destination
- Invokes the user-supplied (or sample) node error program
- Takes the appropriate actions resulting from the defaults which may have been modified by the node error program.

DFHZNAC consists of the following copybooks:

```
DFHZNCA  - Primary error action table and exits
DFHZNCE  - Take action routine
DFHZNCS  - Sense decode routine
DFHZNCV  - VTAM return code routine.
```

DFHZNEP

Entry points: DFHZNENA

Called by: DFHZNAC

Description: The user-replaceable node error program provides:
- A general environment within which it is easy for users to add their own error processors
- Fundamental error recovery actions for a VTAM 3270 network
- The default NEP where the user selects a NEP at system initialization.

DFHZNSP

Entry points: DFHZNSP1

Called by: VTAM

Description: The network service program is invoked when VTAM detects a network service error; for example, when attempting to connect two nodes together, or when the link between two nodes is broken unexpectedly. This module receives control from the VTAM NSEXIT.

DFHZOPA

Entry points: DFHZOPA1

Called by: DFHEIQVT

Description: The open VTAM ACB module is invoked by DFHEIQVT when the master terminal command VTAM OPEN is issued. The ACB is opened and DFHZSLS is called to accept logon requests.

DFHZOPN

Entry points: DFHZOPN1

Called by: DFHZACT

Description: The open destination module acquires storage for an RPL and NIB and BIND areas if the TCTTE does not have these resources already, and sets up the BIND image if required. DFHZOPN then issues a VTAM OPNDST macro (or OPNSEC macro if secondary, to respond to an incoming BIND) to establish a session between CICS and the remote LU.

DFHZOPX

Entry points: DFHZOPX1

Called by: VTAM

Description: The open destination exit module receives control from VTAM on completion of the OPNDST macro in DFHZOPN. If the OPNDST was successful, it indicates in the TCTTE that SDT (start data transfer) is to be sent and checks whether a “good morning” message should be triggered. It then returns to VTAM.

DFHZQUE

Entry points: DFHZQUE1

Called by: All ZCP exits called by VTAM, DFHTCQUE macro

Description: The queue manipulation module processes all requests to add or remove a TCTTE to or from a ZCP activate queue. Additions to the activate queue made by mainline modules use compare-and-swap (CS), because an exit routine may also be adding to the queue asynchronously.

DFHZRAC

Entry points: DFHZRAC1

Called by: DFHZDSP

Description: The receive-any completion module processes the completion of receive-any requests, sets up the TIOA to be passed to attach, and reissues the RECEIVE_ANY macro.

DFHZRAQ

Entry points: DFHZRAQ1

Called by: DFHZARQ, DFHZSYN

Description: The read-ahead queuing module is used to save the inbound data stream in temporary storage when an interlock is caused by both the host and the terminal wanting to send data at the same time.

DFHZRAR

Entry points: DFHZRAR1

Called by: DFHZARQ

Description: The read-ahead retrieval module is called to retrieve data previously saved in temporary storage by DFHZRAQ.

DFHZRAS

Entry points: DFHZRAS1

Called by: DFHZRAC

Description: The receive-any slowdown processing module issues RECEIVE SPEC NQs on LU6.2 sessions for connections and modegroups for which there are ALLOCATE requests queued. This is only done on sessions considered most likely to lead to freeing a “flooding” situation that occurred when LU6.2 connections were reestablished after a failure.

DFHZRLG

Entry points: DFHZRLNA

Called by: DFHZACT

Description: The response logger program logs responses received for protected data sent to an APB. DFHZRLG processes TCTTEs on the log queue when attached by DFHZACT.

DFHZRLP

Entry points: DFHZRLP1

Called by: DFHZDSP

Description: DFHZRLP handles the completion of LU6.2 RECEIVE requests, using the receive RPL addressed by field TCTERPLB in the TCTTE LUC extension. It also manages the logical receive buffer pointers TCTERBLA and TCTERBLL in a consistent manner with the physical receive buffer pointers TCTERBA and TCTERBDL, as (address, length) pairs.

DFHZRLX

Entry points: DFHZRLX1

Called by: VTAM

Description: DFHZRLX is a VTAM exit routine that queues the completed RPL for (post-VTAM) processing by DFHZRLP.

DFHZRRX

Entry points: DFHZRRX1

Called by: VTAM

Description: The release request exit module receives control from VTAM when another application program has requested connection to a terminal currently connected to CICS. If the terminal is not busy, a CLSDST request is queued to the activate chain. Otherwise the release request indicator is set and the request is processed later by module DFHZDET.

DFHZRSP

Entry points: DFHZRSNA

Description: The resynchronization send program performs 3614-dependent actions and is also used to retransmit committed output messages. The message is retrieved from temporary storage if necessary.

DFHZRST

Entry points: DFHZRST1

Called by: DFHZACT, DFHZATI, DFHZCRQ, DFHZDET, DFHZEMW, DFHZERH, DFHZNAC, DFHZRAC, DFHZRSY, DFHZSTU

Description: The RESETSR module changes the mode of a session with a terminal and cancels unsatisfied RECEIVE requests. The mode that is set can be Continue Any (CA) or Continue Specific (CS) and RTYPE=DFSYN, DFASY, or RESP.

DFHZRSY

Entry points: DFHZRSY1

Called by: DFHZACT

Description: The resynchronize module resynchronizes CICS and other nodes of the network. DFHZRSY checks whether inbound and outbound sequence numbers are valid.

DFHZRVL

Entry points: DFHZRVL1

Called by: DFHZARL, DFHZARRL

Description: DFHZRVL processes RECEIVE commands for LU6.2 sessions, using the receive RPL

(RPL_B) addressed by field TCTERPLB in the TCTTE LUC extension. The processing state of the receive RPL is held in the RPL_B state machine field TCTERPBS, also in the TCTTE LUC extension.

DFHZRVS

Entry points: DFHZRVS1

Called by: DFHZACT

Description: The receive specific module initiates a DFSYN receive specific to obtain the next logical record from a node when a user application issues a RECEIVE command.

DFHZRVX

Entry points: DFHZRVX1

Called by: VTAM

Description: The receive specific exit module receives control from VTAM when a receive specific is completed. If the data received is too long for the TIOA provided, the overlength data flag is turned on in the TCTTE and the TCTTE is put back on the activate chain. Otherwise, the response is checked and marked in the TCTTE. The data length is set in the TIOA and the FMH is removed.

DFHZSAX

Entry points: DFHZSAX1

Called by: VTAM

Description: The send DFASY exit module receives control from VTAM when an asynchronous command has completed. It places the TCTTE on the NACP queue if recovery is needed.

DFHZSCX

Entry points: DFHZSCX1

Called by: VTAM

Description: The SCIP exit module is entered whenever the following asynchronous commands are received:
- Non-LU6.2 BIND (as secondary)
- UNBIND (as secondary)
- STSN (as secondary)
- Clear (as secondary)
- SDT (as secondary)
- Request recovery (as primary).

The module correlates BINDs to a TCTTE and schedules DFHZOPN to complete the BIND process. For the other commands, it takes appropriate action and then schedules DFHZNAC using the NACP queue. This module calls DFHZBLX to process LU6.2 binds.

DFHZSDA

Entry points: DFHZSDA1

Called by: DFHZACT, DFHZSDS

Description: The send data flow asynchronous module handles asynchronous command requests. It ensures that an RPL is allocated, primes the RPL for the requested command, and issues the VTAM asynchronous send macro.

DFHZSDL

Entry points: DFHZSDL1

Called by: DFHZARL

Description: DFHZSDL processes SEND commands for LU6.2 sessions, using the RPL addressed by field TCTERPLA in the TCTTE.

DFHZSDR

Entry points: DFHZSDR1

Called by: DFHZACT, DFHZCRQ, DFHZDET, DFHZRVS, DFHZSDA, DFHZSDS

Description: The send response module sends responses to nodes when a synchronization request for a terminal is made and a response is outstanding from a previous operation. If errors occur during task initiation, this module is responsible for the negative response.

DFHZSDS

Entry points: DFHZSDS1

Called by: DFHZACT, DFHZARQ, DFHZATI, DFHZATT, DFHZDET

Description: The send data synchronous module sets up and issues the appropriate VTAM send macro for requests of “send data” or an SNA synchronous command.

DFHZSDX

Entry points: DFHZSDX1

Called by: VTAM

Description: The send data synchronous exit module receives control from VTAM when a SEND request is complete. It checks the RPL for successful completion of the message sent and takes appropriate action.

DFHZSES

Entry points: DFHZSES1

Called by: DFHZACT, DFHZRSY

Description: The session control module is entered whenever a session control command is requested by CICS. It sets up and issues the VTAM SESSIONC command.

DFHZSEX

Entry points: DFHZSEX1

Called by: VTAM

Description: The SESSIONC exit module receives control from VTAM when a SESSIONC command has completed. If the command was successful, it turns off the corresponding flags and enqueues the TCTTE on the activate chain. If the completion was not successful, the TCTTE is placed on the NACP queue for recovery processing.

DFHZSHU

Entry points: DFHZSHU1

Called by: DFHZDSP

Description: The close VTAM ACB module is invoked whenever CICS and VTAM are being uncoupled. This may be as a result of DFHZTPX being driven as the result of a VTAM halt command or the issue of the master terminal command SET VTAM,CLOSE|IMMCLOSE. The status of all sessions is checked and, when all are inactive, the ACB is closed.

DFHZSIM

Entry points: DFHZSIM1

Called by: DFHZACT

Description: The simulate logon module is entered to issue a VTAM SIMLOGON or REQSESS (if secondary) request to place a node in session without the operator having to logon. LU6.2 can be selected by mode name.

DFHZSIX

Entry points: DFHZSIX1

Called by: VTAM

Description: Whenever a SIMLOGON or REQSESS command has been completed, this exit routine is scheduled by VTAM. On successful completion, it turns off the SIMLOGON requested flag and enqueues the TCTTE or TCTME on the activate chain or, if NACP is required, for NACP processing.

DFHZSKR

Entry points: DFHZSKR1

Called by: DFHZACT

Description: The send command response module sends responses to VTAM commands including response to BIND, STSN, and SDT. A positive or negative response can be sent. The module is for secondary LU support only.

DFHZSLS

Entry points: DFHZSLS1

Called by: DFHZDSP, DFHZOPA

Description: The SETLOGON start module issues SETLOGON to cause VTAM to accept automatic logon requests, and issues the initial RECEIVE ANYs for RPLs in the receive-any pool. DFHZSLS also examines the SIT to determine whether autodefine is used. If it is, the appropriate system initialization parameters are copied to the TCT prefix.

DFHZSLX

Entry points: DFHZSLX1

Called by: VTAM

Description: DFHZSLX is a VTAM exit routine that handles the completion of LU6.2 SEND requests.

DFHZSSX

Entry points: DFHZSSX1

Called by: VTAM

Description: The send data flow synchronous exit module receives control when the send of a DFSYN command has been completed.

DFHZSTAP

Entry points: DFHZSTA1

Called by: DFHEGL, DFHETC, DFHETL

Description: DFHZSTAP determines the state of an MRO or LU6.2 conversation from an application viewpoint.

DFHZSTU

Entry points: DFHZSTU1

Called by: DFHTC CTYPE=STATUS, DFHEIQMT, DFHEIQSC, DFHEIQST

Description: DFHZSTU changes the status of TCTTEs and TCTSEs. It can change the following statuses:
- Inservice
- Outservice
- Intlog | No intlog
- Page | Autopage
- ATI | NATI.

DFHZSUP

Entry points: DFHZSUP1

Called by: DFHKCP

Description: The startup task module is the entry point for all terminal-related tasks. DFHZSUP performs the following functions:
- Sets up the TCTTE status
- Performs security checking
- Performs logging of the TCTTE status and input TIOA
- Performs PCT option checking
- Passes control to transaction program, for example, user application, DFHACP, DFHAPRT.

DFHZSYN

Entry points: DFHZSYN1

Called by: DFHDBP

Description: DFHZSYN handles CTYPE=SYNC and RECOVER requests. For protected message support, DFHSPP issues CTYPE=SYNC to clear protected messages. For RECOVER requests, DFHZSYN ensures that no further I/O is issued to that session, and that UNBIND flows.

DFHZSYX

Entry points: DFHZSYX1

Called by: VTAM

Description: The SYNAD exit module receives control from VTAM when a catastrophic error is encountered. DFHZSYX determines the type of error and the appropriate action to be taken, and schedules NACP using the NACP queue to complete the recovery processing.

DFHZTAX

Entry points: DFHZTAX1

Called by: VTAM

Description: The turnaround exit module is called by VTAM on completion of the SEND operation initiated by DFHZRVS in order to perform a turnaround in flip-flop protocol.

DFHZTPX

Entry points: DFHZTPX1

Called by: VTAM

Description: The TPEND exit module receives control when VTAM is terminating. It schedules a CLSDST for each active session if quick shutdown is required, and sets bits in the TCT prefix so that DFHZSHU is invoked.

DFHZTRA

Entry points: DFHZTRA1

Called by: DFHZACT, DFHZDET, DFHZRAC, DFHZRLP, DFHZRVS, DFHZSDL, DFHZSDR, DFHZSDS

Description: DFHZTRA creates VIO trace entries.

DFHZTSP

Entry points: DFHZTSP1

Called by: DFHAPRT, DFHISP, DFHRTE, DFHTPS, DFHZARQ, DFHZCQ, DFHZSUP

Description: The terminal sharing program acquires a TCTTE for a link to a remote CICS address space, and transfers request data to that space. DFHZTSP also receives requests from the remote address space.

DFHZUCT

Entry points: DFHZUCT1

Called by: DFHAPRT, DFHZARQ, DFHZCNA, DFHZRAC, DFHZRVX, DFHZSUP

Description: The uppercase translate module converts a VTAM 3270 data stream into uppercase.

DFHZUIX

Entry points: DFHZUIX1

Called by: DFHZACT, DFHZRAC, DFHZRVX

Description: The user input exit module is called directly (by DFHZRAC) or indirectly (by DFHZRVX via DFHZACT) to link to the user's XZCIN exit.

DFHZUSR

Entry points: DFHZUSR1

Called by: DFHACP, DFHETL, DFHZARER, DFHZARL, DFHZARM, DFHZARR, DFHZARRF, DFHZERH, DFHZOPX, DFHZSTAP, DFHZSUP, DFHZUSR, DFHZXRL, DFHZXRT

Description: DFHZUSR maintains the conversation state for LU6.2.

DFHZXCU

Entry points: DFHZXCU

Description: The VTAM XRF catch-up program is used to send messages that allow a new alternate system to catch up with the current state of the active system for:
- TCT contents
- Bound/unbound state of sessions.

The program is invoked when a new alternate system signs on.

DFHZXQO

Entry points: DFHZXQO

Called by: DFHTCRP, DFHZXST

Description: The XRF ZCP tracking queue organizer allows pending XRF tracking activity to be stored in a way that honors interdependencies, while allowing such requests to be met as soon as all their prerequisites are fulfilled. This component consists of a data structure and accessing program that uses the CICS catalog key structure to identify all the actions for a single resource and the dependencies between them. Actions are put into the structure on receipt in DFHTCRP, and removed by DFHTCRP and at the end of DFHZNAC processing for standby BIND and CLSDST completion. The structure is freed at the end of DFHTCRP tracking.

DFHZXRC

Entry points: DFHZXRC1

Called by: DFHZACT

Description: DFHZXRC analyzes the data received in response to the SESSIONC CONTROL=SWITCH command. It determines the state of the session at the point when it was switched, and initiates the necessary action to clean up and recover the session.

DFHZXRE0

Entry points: DFHZXRE0

Called by: System

Description: DFHZXRE0 runs the CXRE transaction to perform autoconnect and XRF reconnect processing. It also starts the acquire process for terminals with flag TCTEXRE set.

DFHZXRL

Entry points: DFHZXRL1

Called by: DFHZARL, DFHZISP

Description: DFHZXRL is executed in an application-owning region. It routes LU6.2 commands to the terminal-owning region.

DFHZXRT

Entry points: DFHZXRT1

Called by: DFHZTSP

Description: DFHZXRT executes in a terminal-owning region. It receives LU6.2 commands from the application-owning region, and issues them to an APPC device.

DFHZXST

Entry points: DFHZXST

Called by: DFHETC, DFHSIJ1, DFHTCRP, DFHTCRPS, DFHZNAC, DFHZOPA, DFHZXCU

Description: XRF ZCP session-state tracking is called by:
- DFHZNAC for BIND/UNBIND completion in the active system, and for standby-BIND and UNBIND in the alternate system
- DFHETC for logon data freed in the active system
- DFHTCRPS to handle a tracking message
- DFHTCRP to terminate session tracking
- DFHZXCU for BIND/UNBIND catch-up in the active system
- DFHSIJ1 and DFHZOPA to issue a SETLOGON START command.

Part 5. Appendixes

Bibliography

The CICS Transaction Server for z/OS library

The published information for CICS Transaction Server for z/OS is delivered in the following forms:

The CICS Transaction Server for z/OS Information Center

The CICS Transaction Server for z/OS Information Center is the primary source of user information for CICS Transaction Server. The Information Center contains:

- Information for CICS Transaction Server in HTML format.
- Licensed and unlicensed CICS Transaction Server books provided as Adobe Portable Document Format (PDF) files. You can use these files to print hardcopy of the books. For more information, see "PDF-only books."
- Information for related products in HTML format and PDF files.

One copy of the CICS Information Center, on a CD-ROM, is provided automatically with the product. Further copies can be ordered, at no additional charge, by specifying the Information Center feature number, 7014.

Licensed documentation is available only to licensees of the product. A version of the Information Center that contains only unlicensed information is available through the publications ordering system, order number SK3T-6945.

Entitlement hardcopy books

The following essential publications, in hardcopy form, are provided automatically with the product. For more information, see "The entitlement set."

The entitlement set

The entitlement set comprises the following hardcopy books, which are provided automatically when you order CICS Transaction Server for z/OS, Version 3 Release 1:

Memo to Licensees, GI10-2559
CICS Transaction Server for z/OS Program Directory, GI10-2586
CICS Transaction Server for z/OS Release Guide, GC34-6421
CICS Transaction Server for z/OS Installation Guide, GC34-6426
CICS Transaction Server for z/OS Licensed Program Specification, GC34-6608

You can order further copies of the following books in the entitlement set, using the order number quoted above:

CICS Transaction Server for z/OS Release Guide
CICS Transaction Server for z/OS Installation Guide
CICS Transaction Server for z/OS Licensed Program Specification

PDF-only books

The following books are available in the CICS Information Center as Adobe Portable Document Format (PDF) files:

CICS books for CICS Transaction Server for z/OS

General

CICS Transaction Server for z/OS Program Directory, GI10-2586
CICS Transaction Server for z/OS Release Guide, GC34-6421
CICS Transaction Server for z/OS Migration from CICS TS Version 2.3, GC34-6425
CICS Transaction Server for z/OS Migration from CICS TS Version 1.3, GC34-6423
CICS Transaction Server for z/OS Migration from CICS TS Version 2.2, GC34-6424
CICS Transaction Server for z/OS Installation Guide, GC34-6426

Administration

CICS System Definition Guide, SC34-6428

CICS Customization Guide, SC34-6429
CICS Resource Definition Guide, SC34-6430
CICS Operations and Utilities Guide, SC34-6431
CICS Supplied Transactions, SC34-6432

Programming
CICS Application Programming Guide, SC34-6433
CICS Application Programming Reference, SC34-6434
CICS System Programming Reference, SC34-6435
CICS Front End Programming Interface User's Guide, SC34-6436
CICS C++ OO Class Libraries, SC34-6437
CICS Distributed Transaction Programming Guide, SC34-6438
CICS Business Transaction Services, SC34-6439
Java Applications in CICS, SC34-6440
JCICS Class Reference, SC34-6001

Diagnosis
CICS Problem Determination Guide, SC34-6441
CICS Messages and Codes, GC34-6442
CICS Diagnosis Reference, GC34-6899
CICS Data Areas, GC34-6902
CICS Trace Entries, SC34-6443
CICS Supplementary Data Areas, GC34-6905

Communication
CICS Intercommunication Guide, SC34-6448
CICS External Interfaces Guide, SC34-6449
CICS Internet Guide, SC34-6450

Special topics
CICS Recovery and Restart Guide, SC34-6451
CICS Performance Guide, SC34-6452
CICS IMS Database Control Guide, SC34-6453
CICS RACF Security Guide, SC34-6454
CICS Shared Data Tables Guide, SC34-6455
CICS DB2 Guide, SC34-6457
CICS Debugging Tools Interfaces Reference, GC34-6908

CICSPlex SM books for CICS Transaction Server for z/OS

General
CICSPlex SM Concepts and Planning, SC34-6459
CICSPlex SM User Interface Guide, SC34-6460
CICSPlex SM Web User Interface Guide, SC34-6461

Administration and Management
CICSPlex SM Administration, SC34-6462
CICSPlex SM Operations Views Reference, SC34-6463
CICSPlex SM Monitor Views Reference, SC34-6464
CICSPlex SM Managing Workloads, SC34-6465
CICSPlex SM Managing Resource Usage, SC34-6466
CICSPlex SM Managing Business Applications, SC34-6467

Programming
CICSPlex SM Application Programming Guide, SC34-6468
CICSPlex SM Application Programming Reference, SC34-6469

Diagnosis
CICSPlex SM Resource Tables Reference, SC34-6470
CICSPlex SM Messages and Codes, GC34-6471
CICSPlex SM Problem Determination, GC34-6472

CICS family books

Communication

CICS Family: Interproduct Communication, SC34-6473
CICS Family: Communicating from CICS on System/390, SC34-6474

Licensed publications

The following licensed publications are not included in the unlicensed version of the Information Center:
CICS Diagnosis Reference, GC34-6899
CICS Data Areas, GC34-6902
CICS Supplementary Data Areas, GC34-6905
CICS Debugging Tools Interfaces Reference, GC34-6908

Other CICS books

The following publications contain further information about CICS, but are not provided as part of CICS Transaction Server for z/OS, Version 3 Release 1.

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 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

Determining if a publication is current

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

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

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

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

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

C

D

E

F

G

H

I

J

K

L

M

N

O

P

S

T

U

V

W

X

Notices

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

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

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

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

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

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

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

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

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

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft, and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and service names might be trademarks of IBM or other companies.

Sending your comments to IBM

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

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

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

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

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

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

- By mail, to this address:

  IBM United Kingdom Limited
  User Technologies Department (MP095)
  Hursley Park
  Winchester
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–816151
  - From within the U.K., use 01962–816151
- Electronically, use the appropriate network ID:
  - IBMLink: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

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

Product Number: 5655-M15

GC34-6899-03

Spine information:

IBM CICS TS for z/OS CICS Diagnosis Reference Version 3 Release 1