CICS for MVS/ESA

Release Guide

Version 4 Release 1

GC33-1161-04

CICS for MVS/ESA

Release Guide

Version 4 Release 1

GC33-1161-04

Note!

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

Third edition (April 1997)

This edition applies to Version 4 Release 1 of the IBM licensed program Customer Information Control System/Enterprise Systems Architecture (CICS/ESA), program number 5655-018, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Consult the latest edition of the applicable IBM system bibliography for current information on this product.

This is the third edition of the Release Guide for CICS/ESA 4.1. It is based on the second edition, SC33-1161-01, which is now obsolete. Changes from the second edition are marked by the ‘+’ sign to the left of the changes. The vertical lines in the left-hand margins indicate changes made between the first and second editions.

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

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

IBM United Kingdom Laboratories Limited, Information Development,
Mail Point 095, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

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

Contents

Notices

The following paragraph does not apply in any 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 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.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

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 Laboratory Counsel, MP151, IBM United Kingdom Laboratories, 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.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A..

Programming interface information

This book is intended to help you to understand and plan for CICS/ESA Version 4 Release 1.

This book also documents General-use Programming Interface and Associated Guidance Information and Product-sensitive Programming Interface and Associated Guidance Information provided by CICS.

General-use programming interfaces allow the customer to write programs that obtain the services of CICS.

General-use Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or

section or by the following marking:

General-use programming interface

General-use Programming Interface and Associated Guidance Information...

End of General-use programming interface

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of CICS. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Product-sensitive programming interface

Product-sensitive Programming Interface and Associated Guidance Information...

End of Product-sensitive programming interface

Trademarks and service marks

The following terms, used in this publication, are trademarks or service marks of IBM Corporation in the United States or other countries:

Application Development
AD/Cycle
C/370
CICS OS/2
CICS/MVS
CICS/6000
DATABASE 2
DFSMS
Enterprise Systems Architecture/390
ESA/370
GDDM
Hiperbatch
IBMLink
Language Environment
MVS/ESA
MVS/XA
OS/390
RACF
System/390
VTAM
ACF/VTAM
AS/400
CICS
CICS/ESA
CICS/400
CUA
DB2
Enterprise Systems Architecture/370
ES/9000
ESA/390
Hardware Configuration Definition
IBM
IMS/ESA
MVS/DFP
MVS/SP
OpenEdition
PR/SM
RMF
SAA

Preface

What this book is about

This book is about Release 1 of CICS/ESA Version 4, providing introductory, guidance, and reference information. The reference information describes enhancements to:

- The CICS application programming interface (API)
- The exit programming interface (XPI)
- System and resource definition
- Global and task-related user exits
- CICS-supplied transactions, including the master terminal command, CEMT.

Much of the reference information in this book is produced in the same form as it appears in the appropriate reference books. For example, enhancements to the CICS application programming interface (API) commands are shown in the same form as the commands are described in the *CICS/ESA Application Programming Reference*. Changes to this reference information from the CICS/ESA 3.3 editions of the manuals are marked with a vertical line to the left of the changed text.

Note: The programming interface information given in this book is intended only to show what has changed from the previous release of CICS, and to help customers plan their migration to the new release. For definitive programming interface information, refer to the primary sources of programming interface and associated information described. These are the *CICS/ESA Application Programming Reference*, the *CICS/ESA System Programming Reference*, and the *CICS/ESA Customization Guide*.

Who this book is for

This book is for those responsible for the following CICS/ESA 4.1 user tasks:

- Evaluation and planning
- System administration
- Programming
- Customization.

It describes what is new, changed, and obsolete in CICS/ESA 4.1.

What you need to know to understand this book

The book assumes that you are familiar with CICS, either as a systems administrator, or as a systems or application programmer.

How to use this book

This book is organized in the following way:

- In the first five Parts of this book, each chapter describes a major new function introduced in CICS/ESA 4.1. For example, chapter 2 describes transaction isolation, and chapter 3 deals with how CICS has implemented VTAM persistent sessions. These functional descriptions include details of the changes to CICS externals—programming interfaces, resource definitions and so on.

Select and read individual chapters according to your interest in the particular function.

- In Part six, the chapters cover the problem determination changes and improvements introduced by the various items of new function—new messages, details of trace entries, and so on.
- In Part seven, the chapters deal with a wide-ranging series of smaller changes and enhancements.
- Finally, Part eight is a consolidation of the changes to the CICS externals. In these consolidated chapters you can see the overall effect of the changes introduced by different functions on, for example, a single command.

Notes on terminology

"CICS/MVS" is used for Customer Information Control System/Multiple Virtual Storage, and "CICS/ESA" is used for Customer Information Control System/Enterprise System Architecture. Other abbreviations that may be used for CICS releases are as follows:

For CICS/MVS Version 2 Release 1 and subsequent modification levels	– CICS/MVS 2.1
For CICS/ESA Version 3 Release 3	– CICS/ESA 3.3
For CICS/ESA Version 4 Release 1	– CICS/ESA 4.1

The MVS/Enterprise System Architecture (MVS/ESA) operating system, which is a prerequisite for CICS/ESA 4.1, is generally referred to as "MVS."

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 in step, but subsequent updates will probably be available in softcopy before they are available in hardcopy.

For CICS 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-0730-14 is more up-to-date than SK2T-0730-13. The collection kit is also clearly dated on the cover.

Here's how to determine if you are looking at the most current copy of a publication:

- A publication with a higher suffix number is more recent than one with a lower suffix number. For example, the publication with order number SC33-0667-02 is more recent than the publication with order number SC33-0667-01. (Note that suffix numbers are updated as a product moves from release to release, as well as for hardcopy updates within a given release.)
- When the softcopy version of a publication is updated for a new collection kit the order number it shares with the hardcopy version does not change. Also, the date in the edition notice remains that of the original publication. To compare softcopy with hardcopy, and softcopy with softcopy (on two editions of the collection kit, for example), check the last two characters of the publication's filename. The higher the number, the more recent the publication.

For example, DFHPF104 is more recent than DFHPF103. Next to the publication titles in the CD-ROM booklet and the readme files, asterisks indicate publications that are new or changed.

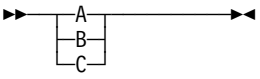
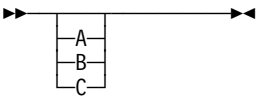
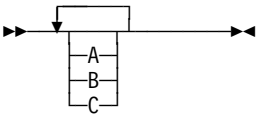
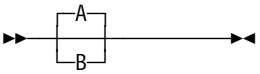
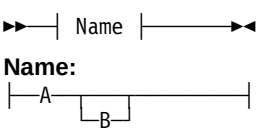
- Updates to the softcopy are clearly marked by revision codes (usually a "#" character) to the left of the changes.

CICS syntax notation used in this book

In the CICS books, CICS commands are presented in a standard way.

The "EXEC CICS" that always precedes each command's keyword is not included; nor is the "END_EXEC" statement used in COBOL or the semicolon (;) used in PL/I and C that you must code at the end of each CICS command. In the C language, a null character can be used as an end-of-string marker, but CICS does not recognize this; you must therefore never have a comma or period followed by a space (X'40') in the middle of a coding line.

You interpret the syntax by following the arrows from left to right. The conventions are:

Symbol	Action
	A set of alternatives—one of which you ***must*** code.
	A set of alternatives—one of which you ***may*** code.
	A set of alternatives—any number of which you may code once.
	Alternatives where **A** is the default.
	Use with the named section in place of its name.
Punctuation and uppercase characters	Code exactly as shown.
Lowercase characters	Code your own text, as appropriate (for example, `name`).

For example, with `READ FILE(filename)` you must code `READ FILE` and `()` unchanged, but are free to code any valid text string to mean the name of the file.

Bibliography

CICS/ESA 4.1 library

Evaluation and planning		
Release Guide	GC33-1161	April 1997
Migration Guide	GC33-1162	April 1997
General		
CICS Family: Library Guide	GC33-1226	April 1995
Master Index	SC33-1187	October 1994
User's Handbook	SX33-1188	April 1997
Glossary (softcopy only)	GC33-1189	n/a
Administration		
Installation Guide	GC33-1163	April 1997
System Definition Guide	SC33-1164	April 1997
Customization Guide	SC33-1165	April 1997
Resource Definition Guide	SC33-1166	April 1997
Operations and Utilities Guide	SC33-1167	April 1997
CICS-Supplied Transactions	SC33-1168	April 1997
Programming		
Application Programming Guide	SC33-1169	October 1994
Application Programming Reference	SC33-1170	April 1997
System Programming Reference	SC33-1171	April 1997
Sample Applications Guide	SC33-1173	October 1994
Distributed Transaction Programming Guide	SC33-1174	October 1994
Front End Programming Interface User's Guide	SC33-1175	October 1994
Diagnosis		
Problem Determination Guide	SC33-1176	October 1994
Messages and Codes	GC33-1177	April 1997
Diagnosis Handbook	LX33-6093	October 1994
Diagnosis Reference	LY33-6082	April 1997
Data Areas	LY33-6083	April 1997
Supplementary Data Areas	LY33-6081	October 1994
Closely-Connected Program Interface	LY33-6084	November 1996
Communication		
Intercommunication Guide	SC33-1181	April 1997
Server Support for CICS Clients	SC33-1591	February 1996
CICS Family: Inter-product Communication	SC33-0824	October 1996
CICS Family: Communicating from CICS/ESA and CICS/VSE	SC33-1697	October 1996
Special topics		
Recovery and Restart Guide	SC33-1182	October 1994
Performance Guide	SC33-1183	October 1994
CICS-IMS Database Control Guide	SC33-1184	October 1994
CICS-RACF Security Guide	SC33-1185	October 1994
Shared Data Tables Guide	SC33-1186	October 1994
External CICS Interface	SC33-1390	April 1997
CICS ONC RPC Feature for MVS/ESA Guide	SC33-1119	February 1996
CICS Web Interface Guide	SC33-1892	November 1996

The book that you are reading was republished in hardcopy format in April 1997 to incorporate updated information previously available only in softcopy. The right-hand column in the above table indicates the latest hardcopy editions of the CICS/ESA books available in April 1997. A book with a date earlier than April 1997 remains the current edition for CICS/ESA 4.1. Note that it is possible that other books in the library will be updated after April 1997.

When a new order is placed for the CICS/ESA 4.1 product, the books shipped with that order will be the latest hardcopy editions.

The style of IBM covers changes periodically. Books in this library have more than one style of cover.

For information about the softcopy books, see “Determining if a publication is current” on page xiv. The softcopy books are regularly updated to include the latest information.

Other CICS books

- *CICS Application Migration Aid Guide*, SC33-0768
- *CICS Application Programming Primer (VS COBOL II)*, SC33-0674
- *CICS/ESA Facilities and Planning Guide* for CICS/ESA Version 3 Release 3, SC33-0654
- *CICS/ESA XRF Guide* for CICS/ESA Version 3 Release 3, SC33-0661
- *CICS Family: API Structure*, SC33-1007
- *CICS Family: General Information*, GC33-0155
- *IBM CICS Transaction Affinities Utility MVS/ESA*, SC33-1159

CICS Clients

- *CICS Clients: Administration*, SC33-1436
- *CICS Family: Client/Server Programming*, SC33-1435

Summary of changes

+ This book is the third edition of the *Release Guide* for CICS for MVS/ESA 4.1.

+ Changes for the third edition

+ This third edition contains changes made as a result of APARs and Reader's
+ Comment Forms. It also contains a new chapter (Chapter 7, “Automatic restart
+ management” on page 113), which became effective following the availability of
+ MVS/ESA 5.2.

+ Changes made for the second edition are still indicated by vertical bars to the left of
+ the changes. Changes made for this third edition are indicated by the ‘+’ symbol to
+ the left of the changes. Users of the second edition can therefore see what has
+ changed since that second edition was published.

+ Softcopy versions of this book also use this revision indicator and also use the ‘#’
+ symbol to show further changes since this third hardcopy edition of the book was
+ published.

| Changes for the second edition

| The second edition was issued at General Availability and contained late changes
| following the early edition that was issued for Announcement of CICS for
| MVS/ESA 4.1 Changes made for this second edition are indicated by the ‘|’ symbol
| to the left of the changes. Users of the first edition can therefore see what
| changed since that first edition was published.

Chapter 1. Summary of CICS/ESA 4.1

This chapter summarizes what is new and changed in CICS/ESA 4.1.

Availability improvements

There are three major items designed to improve availability:

Transaction isolation
This function builds on the CICS storage protection facility, introduced in CICS/ESA 3.3, and provides transaction-to-transaction protection by exploiting the MVS subspace-group facility. Transaction isolation enables you to isolate the user-key task-lifetime storage of a transaction to prevent it being overwritten by user-key programs of other transactions.

See Chapter 2, “Transaction isolation” on page 9 for details, including information about prerequisite hardware and software.

VTAM single-node persistent session support
This function improves the availability of CICS by using the ACF/VTAM 3.4.1 persistent LU-LU session enhancements to enable a fast restart in place in the event of a CICS failure. CICS uses VTAM-retained state data to rebind the CICS sessions.

See Chapter 3, “VTAM persistent sessions” on page 51 for details.

Intersystem session queue management
This function makes it possible to detect that a partner CICS region is under stress by checking the queued ALLOCATE requests. The information passed enables the exit program to determine whether to queue or reject the request. This enables you to avoid further queuing that might result in a slowdown in performance across the interconnected regions.

See Chapter 4, “Intersystem session queue management” on page 71 for details.

MVS sysplex exploitation

The following items are designed to enable CICS to operate more effectively in an MVS sysplex:

Cross-system multiregion operation (XCF/MRO)
This function exploits the cross-system coupling facility (XCF) of MVS/ESA 5.1 to enable MRO to operate across MVS images within a sysplex. The enhanced MRO support also permits new connections to be added while interregion communication (IRC) is open. It also uses an external security manager (such as RACF) instead of CICS internal security mechanisms for MRO security authorization.

See Chapter 5, “Cross-system multiregion operation (XCF/MRO)” on page 87 for details.

Dynamic transaction routing enhancements
These enhancements allow a dynamic transaction routing program to keep track of CICS-initiated transactions, and transactions that abend in an application-owning region (AOR). This function is exploited by the workload balancing routines in CICSPlex SM Release 1.

See Chapter 6, “Dynamic transaction routing enhancements” on page 99 for details.

Intertransaction affinity program offering

This program offering is designed to detect whether application programs potentially cause intertransaction affinity. Transactions that have affinity with one another inhibit the scope for dynamic transaction routing. Detecting intertransaction affinity is the first step in ensuring that workload balancing routines, such as those provided by CICSPlex SM, can manage the affinity.

See Chapter 8, “The IBM CICS Transaction Affinities Utility MVS/ESA” on page 123 for details.

Solution enablement features

The following items, which satisfy some long-standing customer requirements, form a package of major improvements designed to facilitate customer business solutions using CICS:

Non-terminal security

This function supports CICS resource security checking for transactions started without a terminal.

See Chapter 9, “Non-terminal security” on page 127 for details.

New global user exits in the AP domain

There are some new global user exit points added to the CICS EXEC interface layer in the AP domain. These are provided to allow user modification of transient data control, interval control, and temporary storage control commands.

See Chapter 10, “New and enhanced EXEC interface global user exits” on page 137 for details.

External CICS interface

This interface permits a non-CICS program, such as an MVS batch program, to call a CICS application program using CICS distributed program link (DPL) protocols. The called program is invoked as if linked to by another CICS region using DPL.

See Chapter 11, “External CICS interface” on page 169 for details.

Access to CICS state data

This facility provides command-level and exit programming interface (XPI) access to additional CICS state data. It also provides some new exit points, and a programmable interface for purging unwanted started tasks; that is, to purge interval control elements (ICEs).

See Chapter 12, “Access to CICS state data” on page 217 for details.

Cancel start requests

This function enables an application program to purge all the pending transaction start requests associated with a defined terminal or connection. It is provided by a CANCEL option on the EXEC CICS SET TERMINAL and EXEC CICS SET CONNECTION commands.

See Chapter 13, “Cancel start requests” on page 239 for details.

System management improvements

There are five items designed to improve system management:

Autoinstall for programs, mapsets, and partitionsets

This extends the CICS autoinstall facility to enable CICS to install programs, mapsets, and partitionsets without prior definition of each resource. This autoinstall facility is similar to that already supported for terminals, and is based on customer requirements.

The autoinstall services for these additional resources are provided by the new program manager domain.

See Chapter 14, “Autoinstall for programs, mapsets, and partitionsets” on page 249 for details.

Autoinstall for LU6.2 parallel sessions

This extends the CICS autoinstall facility, enabling CICS to install LU6.2 parallel sessions automatically without the need for every session to be explicitly defined in the CICS system definition (CSD) data set.

See Chapter 15, “Autoinstall for APPC connections” on page 265 for details.

CICS support for the MVS workload manager

This support provides CICS data to the MVS/ESA 5.1 workload manager. The MVS workload manager provides a goal-oriented, self-tuning capability that improves overall system performance.

See Chapter 16, “CICS/ESA support for MVS workload management” on page 275 for details.

Global install and discard exit

This new global user exit is driven whenever a CICS resource definition is installed or discarded. It allows a system management program to keep track of which CICS resources are installed on which CICS systems. It is used by CICSPlex SM.

See Chapter 17, “Install and discard global user exit” on page 291 for details.

Monitoring and statistics enhancements

These enhancements to CICS monitoring and statistics data derive from the many changes and enhancements in CICS/ESA 4.1. For example, some are a result of the incorporation of the front-end programming interface (FEPI) into CICS/ESA 4.1, and some are part of the CICS support for MVS workload management.

See Chapter 18, “Monitoring and statistics” on page 295 for details.

CICS internal restructure

CICS reliability and availability are further enhanced by the continued restructure of CICS code and control blocks that began in CICS/ESA Version 3. There are five new domains introduced in CICS/ESA 4.1.

Directory manager domain

This domain provides, for other domains, the look-up services for transactions, transaction classes, programs, mapsets, partitionsets, and user attributes. It replaces the function previously provided by the CICS table manager program for these resources.

See Chapter 19, “Directory manager domain” on page 327 for details.

Program manager domain
This domain provides program control services for the management of user application programs. It replaces the function previously provided by the CICS program control program.

See Chapter 20, “Program manager domain” on page 331 for details.

Transaction manager domain
This domain provides services to create, terminate, purge, and inquire about CICS tasks. It also provides a transaction environment to enable other CICS components implement transaction-related services.

See Chapter 21, “Transaction manager domain” on page 341 for details.

Security manager domain
This domain provides the security services required for all CICS security checking. This domain handles CICS resource security, CICS command security, and CICS surrogate-user security, and all interfaces with the external security manager (ESM).

See Chapter 22, “Security manager domain” on page 385 for details.

User domain
This domain manages the state data of all end-users of CICS. The user state data includes information such as userid, operator identification, operator class, operator priority and so on. The data managed by the user domain replaces entirely the CICS sign-on table of earlier releases, which is now obsolete.

See Chapter 23, “User domain” on page 401 for details.

Signon
The signon component of terminal control is restructured, in parallel with the user domain and security manager domain restructure. The restructure includes a new signon authorization process, an enhanced CICS-supplied signon transaction, and a new timeout transaction.

See Chapter 24, “Signon component” on page 407 for details.

Problem determination

Changes and enhancements in CICS/ESA 4.1 improve problem determination, and include many new trace points and messages. These are described in the following chapters:

- Chapter 25, “Problem determination changes” on page 423
- Chapter 26, “CICS dumping in a sysplex” on page 427

General enhancements

There are many new items of a more minor nature in CICS/ESA 4.1, in addition to the main features described above.

Database management enhancements
These enhancements to CICS database support relate mainly to changes in the CICS interface with IMS, and particularly with DBCTL. There are also changes to the CICS DB2 attachment facility.

See Chapter 27, “Changes to database management” on page 441 for details.

Changes for application programming
This chapter describes a number of small enhancements for application programming.

See Chapter 28, “Changes for application programming” on page 463 for details.

Changes for security
This chapter describes some enhancements for CICS security.

See Chapter 29, “Changes for security” on page 479 for details.

Changes for sysplex exploitation
This chapter describes some enhancements for CICS in the sysplex environment.

See Chapter 30, “Changes for sysplex exploitation” on page 487 for details.

Changes for system and resource definition
This chapter describes some enhancements in CICS/ESA 4.1 that affect system and resource definition.

See Chapter 31, “Changes for system and resource definition” on page 499 for details.

Changes for installation, operations, and customization
This chapter describes a number of changes in CICS/ESA 4.1 that affect installation, operations, and customization.

See Chapter 32, “Changes for installation, operations, and customization” on page 529 for details.

Consolidated externals

Some of the major new functional items, described in their individual chapters, cause changes to the same commands or resource definitions. The final chapters in the book consolidate all the individual changes so that you can see the effect on command syntax as a whole.

Part 1. Availability improvements

This Part describes availability improvements provided in CICS/ESA 4.1, under the following topics:

Chapter 2. Transaction isolation

This chapter describes the transaction isolation facility of CICS/ESA 4.1. It covers the following topics:

- Overview
- Benefits of transaction isolation
- Requirements for transaction isolation
- Changes to CICS externals
- Problem determination.

Overview

Transaction isolation in CICS/ESA 4.1 offers storage protection between transactions, ensuring that a program of one transaction does not accidentally overwrite the storage of another transaction. Accidental overwrites of the transaction data of an application program can affect the reliability and availability of a CICS system, and the integrity of data in the system.

Transaction isolation extends the storage protection introduced in CICS/ESA 3.3, which protected CICS system storage from being accidentally overwritten by user-written application programs.

To provide protection for the transaction data of application programs, the CICS transaction isolation facility requires hardware and MVS facilities that support subspace groups (see “Requirements for transaction isolation” on page 17 for details).

New problem determination features allow an errant application program to be identified as soon as:

- The program itself attempts to access a storage area it doesn't own, or
- The program passes a bad address to CICS, which would cause CICS to perform the storage overwrite in situations where the application itself cannot do so because of storage protection mechanisms.

Previously, it was time-consuming, if not impossible, to identify rogue application programs because usually they have ceased execution some time before the failure resulting from the overwrite occurs, and often are no longer resident in the CICS address space.

Preventing programs passing bad addresses to CICS should result in fewer calls to IBM service resulting from ambiguous ‘user errors’. Previously when CICS failed from an overwrite, the service center could be called as it appeared to be a CICS system problem. In CICS/ESA 4.1, an application storage protection exception condition is identified as an application problem, saving both the user's and IBM's time.

Transaction isolation concepts

CICS storage protection, introduced in CICS/ESA 3.3, uses the extension to MVS key-controlled storage protection that is provided by the subsystem storage protection facility. CICS storage protection is controlled by programs being defined to run in either user key or CICS key:

Storage key	Description
User key	defines both the storage and execution key for user application programs, as follows: • The key of the storage that user applications and their data areas normally reside in, allocated in user dynamic storage areas above or below 16MB. • The execution (PSW) key that application programs normally run with. This gives user-key application programs read and write access to user-key storage, but read-only access to CICS-key storage.
CICS key	defines both the storage and execution key for CICS programs, as follows: • The key of the storage that CICS code and data areas normally reside in, allocated in CICS dynamic storage areas above or below 16MB. • The execution (PSW) key that CICS runs with. This gives CICS read/write access to both CICS-key and user-key storage. (CICS key is the key in which CICS is given control by MVS at job step initiation—key 8.)

CICS transaction isolation builds on CICS storage protection, enabling user transactions also to be protected from one another. In CICS/ESA 4.1, in addition to being able to specify the storage key for transactions and the execution key for programs, you can specify whether you want transaction isolation. For individual transactions you do this on the ISOLATE option of the transaction resource definition.

You can also control transaction isolation globally for the whole CICS region by means of the TRANISO system initialization parameter.

MVS subspaces

MVS/ESA 5.1 introduces the subspace group facility, which can be used for storage isolation to preserve data integrity within an address space.

The subspace group facility uses new hardware to provide protection for transaction data. A subspace group is a group of subspaces and a single base space, where the base space is the normal MVS address space as in releases of MVS/ESA prior to MVS/ESA 5.1.

The subspace group facility provides a partial mapping of the underlying base space, so that only specified areas of storage in the base space are exposed in a particular subspace. Thus each subspace represents a different subset of the storage in the base space. Transaction isolation, when specified, ensures that programs defined with EXECKEY(USER) execute in their own subspace, with appropriate access to any shared storage, or to CICS storage. Thus a user

transaction is limited to its own ‘view’ of the address space. See Figure 1 on page 15 for an illustration of the subspace group concept.

Programs defined with EXECKEY(CICS) execute in the base space, and have the same privileges as in CICS/ESA 3.3.

Subspaces for user transactions

In general, transaction isolation ensures that user-key programs are allocated to separate (unique) subspaces, and have:

- Read and write access to the user-key task-lifetime storage of their own tasks, which is allocated from one of the user dynamic storage areas (UDSA or EUDSA)
- Read and write access to shared storage; that is, storage obtained by EXEC CICS GETMAIN commands with the SHARED option (SDSA or ESDSA)
- Read access to the CICS-key task-lifetime storage of other tasks (CDSA or ECDSA)
- Read access to CICS code
- Read access to CICS control blocks that are accessible via the CICS API.

They do not have any access to user-key task-lifetime storage of other tasks.

The defaults for new transaction resource definition attributes specify that existing application programs operate with transaction isolation (the default for the ISOLATE option is YES). Existing applications should run unmodified provided they conform to transaction isolation requirements.

However, a minority of applications may need special definition if they:

- Issue MVS macros directly, or
- Modify CICS control blocks, or
- Have a legitimate need for one task to access, or share, another task's storage.

Some existing transactions may share task-lifetime storage, in various ways, and this sharing may prevent them running isolated from each other. To allow such transactions to continue to execute, without requiring that they run in the base space (where they could corrupt CICS data or programs) a single common subspace is provided in which all such transactions can execute. They are then isolated from the other transactions in the system that are running in their own subspaces, but able to share each other's data within the common subspace. See “The common subspace and shared storage” on page 12 for more information.

To help you identify transactions that share task-lifetime storage, IBM provides a program offering, the IBM CICS Transaction Affinities Utility MVS/ESA. See Chapter 8, “The IBM CICS Transaction Affinities Utility MVS/ESA” on page 123 for an overview of this program offering.

Selecting transaction isolation

Transaction isolation is a global option that you select by a CICS system initialization parameter, and works in conjunction with the CICS storage protection introduced in CICS/ESA 3.3.

At the individual transaction level, resource definition parameters (such as EXECKEY, TASKDATAKEY, and ISOLATE) allow you to specify the level of protection that you want for each transaction and program. The defaults for these options mean that in most cases no changes to definitions are needed for existing applications. However, where needed, protection can be tailored to allow transactions to continue to function where they fail to meet the criteria for full protection, which is the default.

Defining CICS dynamic storage areas

It is no longer necessary to specify CICS dynamic storage area (DSA) and storage cushion sizes on system initialization parameters.

The number of DSAs has increased from five to eight. The new DSAs in this release are:

- The shared dynamic storage area (SDSA)
- The extended shared dynamic storage area (ESDSA)
- The read-only dynamic storage area (RDSA).

You do not need to specify and tune the sizes of the DSAs and their storage cushions; CICS dynamically manages the individual DSA and cushion sizes automatically, to optimize storage usage. Also, you no longer need to restart CICS to tune these areas enabling CICS regions to operate continuously for longer than in previous releases.

+ **APAR PN70228**

+ Documentation for PN70228 added on 27 September 1996

+ Two new system initialization parameters, DSALIM and EDSALIM, set the overall
+ dynamic storage limits both above and below 16MB. Within these overall storage
+ limits, CICS controls the individual DSA sizes and their associated storage
+ cushions.

You can dynamically modify the overall storage limits for DSAs using the CEMT SET SYSTEM or EXEC CICS SET SYSTEM commands. You cannot modify individual DSA sizes or their storage cushions. Care should be taken when sizing DSALIM and EDSALIM. A good knowledge of storage requirements outside the dynamic areas is required to ensure that there is sufficient MVS storage for other subsystem use.

The common subspace and shared storage

You might have some transactions where the application programs access one another's storage in a valid way. One such case is when a task waits on one or more event control blocks (ECBs) that are later posted, either by an MVS POST or ‘hand posting’, by another task.

For example, a task can pass the address of a piece of its own storage to another task (via a temporary storage queue or some other method) and then WAIT for the other task to post an ECB to say that it has updated the storage. Clearly, if the original task is executing in a unique subspace, the posting task will fail when attempting the update and to post the ECB, unless the posting task is executing in CICS key. CICS therefore checks when a WAIT is issued that the ECB is in shared storage, and raises an INVREQ condition if it is not. ECBs need to reside below 16MB, that is, in the SDSA. Shared storage is allocated from one of the user-key shared dynamic storage areas, below or above 16MB (SDSA or ESDSA).

CICS/ESA 4.1 supports the following methods to ensure that transactions that need to share storage can continue to work in the subspace group environment:

- You can specify that all the related transactions are to run in the common subspace. The common subspace allows tasks that need to share storage to coexist, while isolating them from other transactions in the system. Transactions assigned to the common subspace have the following characteristics:
  - They have read and write access to each other's task-lifetime storage
  - They have no access of any kind to storage of transactions that run in unique subspaces
  - They have read access only to CICS storage

  Any group of related transactions that work in user key in CICS/ESA 3.3 should work under CICS/ESA 4.1 if defined with ISOLATE(NO) to ensure they run in the common subspace. This provides support for migration, allowing the separation of transactions into their own unique subspaces to be staged gradually after installing CICS/ESA 4.1 and related support.
- You can ensure that the common storage is in SHARED storage by obtaining the storage with the SHARED option.
- You can ensure that the application programs of the transactions that are sharing storage are all defined with EXECKEY(CICS). This ensures that their programs execute in base space, where they have read/write access to all storage. However, this method is not recommended because it does not give any storage protection.

Required base space usage

With the exception of some global and task-related user exits, programs that execute in CICS key run in base space. There is no mechanism to make a CICS-key program run in a subspace.

Global and task-related user exits: These must all execute in CICS key and even if defined with EXECKEY(USER), CICS enforces CICS-key execution. However, in a CICS region with transaction isolation active, they may run in either base space or a subspace, depending on the current mode when CICS gives control to the exit program.

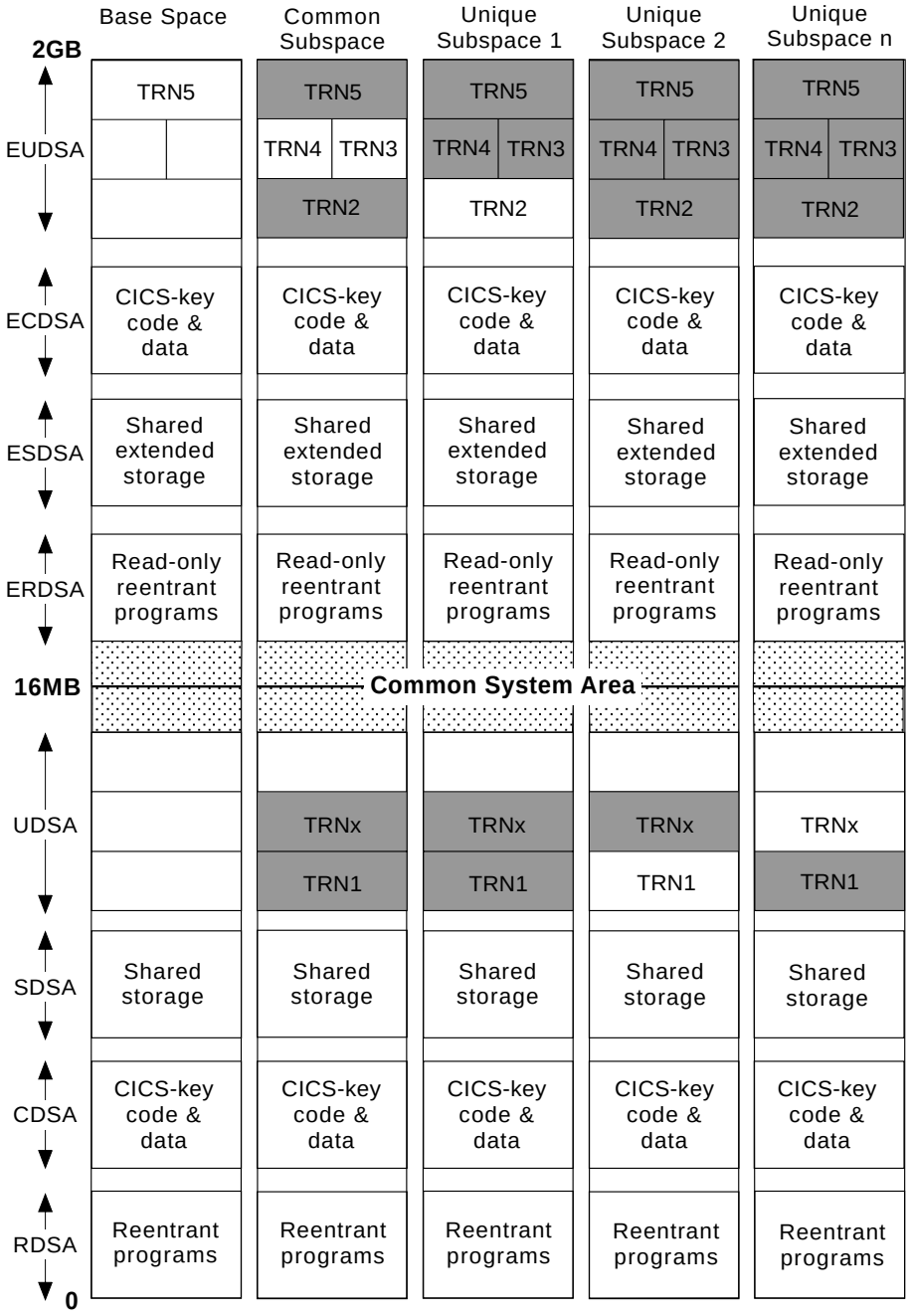
Note that if you have a valid reason why a global user exit program should run in base space, there is an option provided on the exit programming interface (XPI) that allows you to switch from a subspace into base space. See “Changes to the exit programming interface” on page 39 for details of this new XPI function.

User replaceable modules: These must all execute in CICS key and even if defined with EXECKEY(USER), CICS enforces CICS key execution. They run in base space and are able to access all storage in the CICS address space and, like CICS itself, can modify storage belonging to any application program.

Program list table (PLT) programs: Initialization PLT programs run under control of a CICS task in CICS key, and therefore execute in base space. Their storage is in CICS-key storage, and protected from overwriting by other transactions.

Termination PLT programs, which can run under a user task that issues EXEC CICS PERFORM SHUT, also execute in base space and in CICS key.

Transaction isolation concepts are illustrated in Figure 1 on page 15.



Notes:

1. Programs executing in the base space can 'see' all the storage in the CICS address space (base space=address space). In this illustration, TRN5 is a user task running in CICS key and therefore is executing in the base space.
2. The shaded areas in the common subspace and in the unique subspaces are 'hidden' areas of storage owned by user tasks. The areas are exposed (addressable) only to tasks in their own subspace (and, of course, to tasks in the base space).

Figure 1. Some views of one CICS address space with transaction isolation active

Benefits of transaction isolation

Some of the benefits of transaction isolation, and its associated support, are:

- Reducing system outages
- Protecting application data
- Protecting CICS from application programs that pass invalid addresses
- Simplifying the definition of storage sizes for the eight CICS dynamic storage areas
- Aiding application development.

Reducing system outages

Transaction isolation prevents unplanned CICS system outages caused by coding errors in user-key application programs that cause the storage of user-key transactions to be accidentally overwritten. Prevention of accidental transaction data overwrites significantly improves the reliability and availability of CICS regions.

Protecting application data

If an application program overwrites CICS code or data, CICS can fail as a result. If an application program overwrites another application program's **code** then that other application program is likely to fail. While this is a serious interruption in a production region, the effect is immediate and the program can generally be recovered so that the terminal user can retry the failed transaction. However, if an application program overwrites the **data** of another application program, the results often are not immediately apparent; the erroneous data can be written to a database and the error remain undetected until later, when it may be impossible to determine the cause of the error. The consequences of a data overwrite are often much more serious than a code overwrite.

Protecting CICS from being passed bad addresses

CICS/ESA 4.1 also protects against applications that pass bad addresses to CICS that would result in storage violations caused by CICS. This occurs when an application program issues an EXEC CICS command that causes CICS to modify storage on the program's behalf, but the program does not own the storage. In earlier releases, CICS did not check ownership of the storage referenced by the passed address, and executed such commands with consequent storage violations.

In CICS/ESA 4.1, CICS validates the start address of the storage, and establishes that the application program has write access to the storage that begins with that address, before executing the command.

Simplifying the definition of CICS dynamic storage areas

+ The allocation of CICS dynamic storage areas can now be done automatically. This removes the need to specify the size of each individual dynamic storage area. You need specify only the overall limits within which CICS can allocate storage for these areas.

CICS now uses eight separate dynamic storage areas. The sizes of the five DSAs used in CICS/ESA 3.3 are specified on individual system initialization parameters, the dynamic storage areas then being fixed for the whole of the CICS execution.

To facilitate continuous operations, and also to simplify CICS/ESA system management, the individual DSA sizes can be determined by CICS, and can be varied dynamically by CICS as the need arises. You simply specify how much storage that CICS is to use for the DSAs in two amounts—one for the four DSAs above 16MB, and the other for the four DSAs below. Automatic sizing within the specified limits removes the need for restarts to change DSA sizes. You can also vary the overall limits dynamically, using either the CEMT master terminal command, or an EXEC CICS SET command.

Aiding application development

Transaction isolation aids application development in the testing and debugging phase. If an application tries to overwrite CICS or another application, or if it tries to pass a storage address it doesn't own for CICS to write to, CICS immediately abends the task and reports the rogue program's name and the area it tried to overwrite. This saves much time trying to debug what is a common problem in application development environments.

Requirements for transaction isolation

For CICS transaction isolation support you need the following software:

- MVS/ESA SP – JES2 Version 5 Release 1 or a later, upward-compatible, release
- MVS/ESA SP – JES3 Version 5 Release 1 or a later, upward-compatible, release

JES2 and JES3 release levels: As an alternative to using the JES2 or JES3 component supplied with MVS/ESA SP 5.1, you can use the MVS/ESA SP 5.1 Base Control Program (BCP) with the JES2 or JES3 component provided with an earlier release of MVS/ESA SP.

In addition to the prerequisite release of MVS, you need an Enterprise Systems Architecture/390 (ESA/390) processor that includes the subspace-group facility. See the latest IBM hardware information for datails of machines that support subspace-groups.

Resource usage

You should review the region size specified on the REGION parameter for CICS address spaces. The increase in CICS use of virtual storage above 16MB means that you will probably need to increase the REGION parameter.

The introduction of the transaction isolation facility increases the allocation of some virtual storage above 16MB for CICS regions that are running with transaction isolation active.

If you are running with transaction isolation active, CICS allocates storage for task-lifetime storage in multiples of 1MB for user-key tasks that run above 16MB. (1MB is the minimum unit of storage allocation above 16MB for the EUDSA when transaction isolation is active.) However, although storage is allocated in multiples of 1MB above 16MB, MVS paging activity affects only the storage that is actually used (referenced), and unused parts of the 1MB allocation are not paged.

If you are running without transaction isolation, CICS allocates user-key task-lifetime storage above 16MB in multiples of 64KB.

The subspace group facility uses more real storage because MVS creates, for each subspace, a page and segment table from real storage. The CICS/ESA 4.1 requirement for real storage varies according to the transaction load at any one time. As a guideline, each task in the system requires 9KB of real storage, and this should be multiplied by the number of concurrent tasks that can be in the system at any one time (governed by the MXT system initialization parameter).

However, automatic DSA sizing removes the need for accurate storage estimates, with CICS dynamically changing the size of DSAs as demand requires.

Changes to CICS externals

The introduction of transaction isolation and the changes to storage management result in a number of changes to CICS externals. These are:

- Changes to system definition
- Changes to resource definition
- Changes to the application programming interface (API)
- Changes to the system programming interface (SPI)
- Changes to global user exits
- Changes to the exit programming interface (XPI)
- Changes to user replaceable modules
- Changes to CICS-supplied transactions

Changes to system definition

There are additions to CICS system initialization parameters in CICS/ESA 4.1 to support transaction isolation, and some parameters of CICS/ESA 3.3 are now obsolete. The obsolete parameters are as follows:

CSCS
ECSCS
ERSCS
EUSCS
USCS

+ The system initialization parameters that are new or changed for transaction
+ isolation are shown in Table 1. For details of all the system initialization
parameters available in CICS/ESA 4.1 see the *CICS/ESA System Definition Guide*.

+ **APARs PN70228 & PN 88030**

+ The following table changed for PN70228 (27 September 1996) and changed
+ for PN88030 (21 May 1997).

Table 1. The DFHSIT macro parameters

	DFHSIT	[TYPE={**CSECT**|DSECT}]
		⋮
+		[,CDSASZE={**0K**|number}]
+		[,CMDPROT={**YES**|NO}]
		⋮
		[,DSALIM={**5M**|number}]
		⋮
|		[,EDSALIM={**20M**|number}]
#		[,ECDSASZE={**0M**|number}]
#		[,ERDSASZE={**0M**|number}]
#		[,EUDSASZE={**0M**|number}]
		⋮
+		[,RDSASZE={**0K**|number}]
+		[,SDSASZE={**0K**|number}]
+		[,UDSASZE={**0K**|number}]
|		[,TRANISO={**NO**|YES}]
		⋮
	END	DFHSITBA

Note: The following parameter descriptions do not include the individual DSA size
parameters, which were reinststed by APARs PN70228 and PN 88030.
The changes to these system initialization parameters were in the default
values, which became 0 (zero), meaning that the DSA can be changed
dynamically by CICS, whereas a non-zero value indicates that the DSA size
is fixed.

CMDPROT={YES|NO}

Code this parameter to allow, or inhibit, CICS validation of start addresses of storage referenced as output parameters on EXEC CICS commands.

YES

If you specify YES, CICS validates the initial byte at the start of any storage that is referenced as an output parameter on EXEC CICS commands to ensure that the application program has write access to the storage. This ensures that CICS does not overwrite storage on behalf of the application program when the program itself cannot do so. If CICS detects that an application program has asked CICS to write into an area to which the application does not have addressability, CICS abends the task with an AEYD abend.

The level of protection against bad addresses depends on the level of storage protection in the CICS environment. The various levels of

protection provided when you specify CMDPROT=YES are shown in Table 3 on page 31.

NO
If you specify NO, CICS does not perform any validation of addresses of the storage referenced by EXEC CICS commands. This means that an application program could cause CICS to overwrite storage to which the application program itself does not have write access.

DSALIM={5M|number}
Specifies the upper limit of the total amount of storage within which CICS can allocate the individual dynamic storage areas (DSAs) that reside below 16MB.

5M
The default DSA limit is 5MB (5 242 880).

number
Specify *number* as an amount of storage in the range 2MB to 16MB (2 097 152 bytes to 16 777 216 bytes) in multiples of 262 144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify *number* in bytes (for example, 4 194 304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

From the storage size that you specify on the DSALIM parameter, CICS allocates the following dynamic storage areas:

The user DSA (UDSA)
The user-key storage area for all user-key task-lifetime storage below 16MB.

The read-only DSA (RDSA)
The key-0 storage area for all reentrant programs and tables below 16MB.

The shared DSA (SDSA)
The user-key storage area for any non-reentrant user-key RMODE(24) programs, and also for any storage obtained by programs issuing EXEC CICS GETMAIN commands for storage below 16MB with the SHARED option.

The CICS DSA (CDSA)
The CICS-key storage area for all non-reentrant CICS-key RMODE(24) programs, all CICS-key task-lifetime storage below 16MB, and for CICS control blocks that reside below 16MB.

Notes:

1. CICS allocates the UDSA in multiples of 1MB when transaction isolation is active, but in multiples of 256KB in CICS regions without transaction isolation. The other DSAs below 16MB are allocated in multiples of 256KB, with or without transaction isolation. The maximum you can specify depends on a number of factors, such as how you have configured your MVS storage (which governs how much private storage remains below 16MB) and how much private storage you must leave free to satisfy MVS GETMAIN requests for storage outside the DSAs.

2. For information about calculating the amount of storage to specify on the DSALIM parameter, see the *CICS/ESA Performance Guide*.

EDSALIM={20M|number}

Specifies the upper limit of the total amount of storage within which CICS can allocate the individual extended dynamic storage areas (EDSAs) that reside above 16MB.

20M

The default EDSA limit is 20MB (20 971 520 bytes).

number

Specify *number* as a value in the range 10MB to 2047MB, in multiples of 1MB. If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify *number* in bytes (for example, 33 554 432), or as a whole number of kilobytes (for example, 32 768K), or a whole number of megabytes (for example, 32M).

The maximum value allowed depends on a number of factors, such as:

- The size of the region you have specified on the MVS REGION parameter in the CICS job or procedure.
- How much storage you require for the CICS internal trace table.
- How much private storage you must leave free to satisfy MVS GETMAIN requests for storage above 16MB outside the DSAs.

From the storage value that you specify on the EDSALIM parameter, CICS allocates the following extended dynamic storage areas:

The extended user DSA (EUDSA)

The user-key storage area for all user-key task-lifetime storage above 16MB.

The extended read-only DSA (ERDSA)

The key-0 storage area for all reentrant programs and tables above 16MB.

The extended shared DSA (ESDSA)

The user-key storage area for any non-reentrant user-key RMODE(ANY) programs, and also for any storage obtained by programs issuing CICS GETMAIN commands for storage above 16MB with the SHARED option.

The extended CICS DSA (ECDSA).

The CICS-key storage area for all non-reentrant CICS-key RMODE(ANY) programs, all CICS-key task-lifetime storage above 16MB, and CICS control blocks that reside above 16MB.

CICS allocates all the DSAs above 16MB in multiples of 1MB.

Note: For information about calculating the amount of storage to specify on the EDSALIM parameter, see the *CICS/ESA Performance Guide*. For example, the value that you specify must allow for all temporary storage MAIN requests, for which CICS uses the ECDSA.

TRANISO={NO|YES}

Code this parameter, together with the STGPROT system initialization parameter, to specify whether you want transaction isolation in the CICS region. The permitted values are NO (the default), or YES:

NO

This is the default. If you specify NO, or allow this parameter to default, CICS operates without transaction isolation, and all storage in the CICS address space is addressable as in earlier releases. If you specify STGPROT=YES and TRANISO=NO, CICS storage protection is active without transaction isolation.

YES

Specify YES for transaction isolation. If you specify TRANISO=YES and STGPROT=YES, and you have the required hardware and software, CICS operates with transaction isolation. This ensures that the user-key task-lifetime storage of transactions defined with the ISOLATE(YES) option is isolated from the user-key programs of other transactions.

If you specify TRANISO=YES, but you do not have the required hardware and software or STGPROT=NO is specified, CICS issues an information message during initialization, and operates without transaction isolation.

STGPROT=NO and TRANISO=YES specified in the system initialization table causes an error during assembly (MNOTE 8).

Changes to resource definition

There are some changes to CICS resource definition to support transaction isolation, as follows:

- For transaction resource definitions there is a new ISOLATE parameter, and the description of the TASKDATAKEY attribute is changed
- For program resource definitions the description of the EXECKEY parameter is changed.

Transaction resource definition changes

A new attribute, ISOLATE, is added to the transaction resource definition, and the definition of the TASKDATAKEY option is affected by the introduction of the ISOLATE option.

```
 TRansaction  ==> ....
 Group        ==> ........
 DEscription  ==> ............................................
 PROGram      ==> ........
 TWasize      ==> 00000             0-32767
 PROFile      ==> DFHCICST
 PArtitionset ==> ........
 ...
 ISOLATE      ==> Yes               Yes | No
 TASKDATAKey  ==> User              User | CICS
 TASKDAtaloc  ==> Below             Below | Any

REMOTE ATTRIBUTES
 DYnamic      ==> No                No | Yes
 ...
 ...
```

Figure 2. The DEFINE panel for the TRANSACTION resource definition

ISOLATE({YES|NO})
indicates whether CICS is to isolate the transaction's user-key task-lifetime storage to provide transaction-to-transaction protection. (See the TASKDATAKEY option for a description of user-key storage.) Isolate means that the user-key task-lifetime storage is protected from both reading and writing by the user-key programs of other transactions—that is, from programs defined with EXECKEY(USER).

Note: The ISOLATE option does not provide any protection against application programs that execute in CICS key—that is, from programs defined with EXECKEY(CICS).

YES
If you specify ISOLATE(YES), the transaction's user-key task-lifetime storage is isolated from the user-key programs of all other transactions—that is, from programs defined with EXECKEY(USER), but not from programs defined with EXECKEY(CICS).

Also, the user-key task-lifetime storage of **all** other transactions is protected **from** the user-key programs of transactions defined with ISOLATE(YES).

NO
If you specify ISOLATE(NO), the transaction's task-lifetime storage is isolated from the user-key programs of those transactions defined with ISOLATE(YES). The transaction's storage is not, however, isolated from user-key programs of other transactions that also specify ISOLATE(NO), because with this option, the transactions are all allocated to the common subspace.

Note also that the user-key task-lifetime storage of all transactions defined with ISOLATE(YES) is protected **from** the user-key programs of transactions defined with ISOLATE(NO).

You should specify ISOLATE(NO) for those transactions that require to share any part of their user-key task-lifetime storage.

The ISOLATE option is illustrated in Figure 3 on page 24.

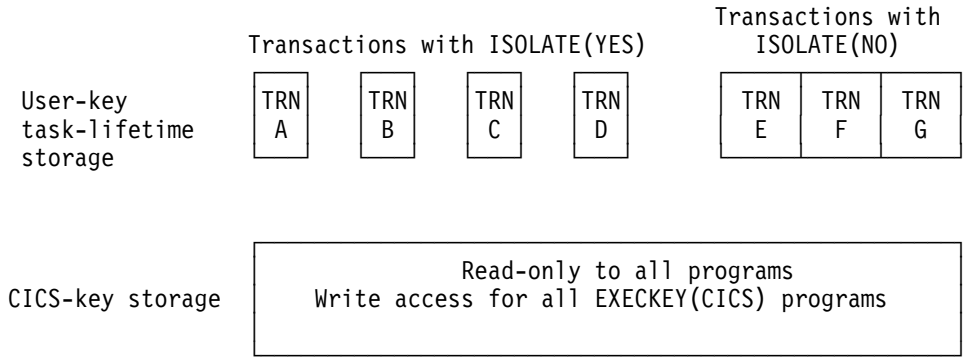


Figure 3. Conceptual view of transactions with and without transaction isolation. In this diagram, the elements of user-key task-lifetime storage of each of the transactions A, B, C, and D are isolated from each other, and from transactions E, F, and G. For the transactions A, B, C and D, therefore, their storage is addressable by their own user-key programs only.

The user-key task-lifetime storage of transactions E, F, and G, however, is accessible by all the user-key application programs of transactions E, F and G, but is isolated from the user-key programs of transactions A, B, C, and D. All of the illustrated transactions have read-only access to CICS-key storage.

TASKDATAKEY(USER|CICS)

This specifies the storage key of the storage CICS allocates at task initialization for the duration of the task (task-lifetime storage), and which is accessible by the application. These storage areas are the EXEC interface block (EIB) and the transaction work area (TWA).

TASKDATAKEY also specifies the key of the storage that CICS obtains on behalf of all programs that run under the transaction. The program-related storage that CICS allocates in the specified key includes:

- The copies of working storage that CICS obtains for each execution of an application program
- The storage CICS obtains for the program in response to implicit and explicit GETMAIN requests. For example, the program can request storage by a GETMAIN command, or as a result of the SET option on other CICS commands.

You must specify TASKDATAKEY(USER) if any of the programs in the transaction are defined with EXECKEY(USER). If you specify TASKDATAKEY(CICS) for a transaction, an attempt to run any program in user key under this transaction leads to a task abend, with abend code AEZD.

USER

CICS obtains user-key storage for this transaction. Application programs executing in any key can both read and modify these storage areas.

Note: User-key programs of transactions defined with ISOLATE(YES) have access only to the user-key task-lifetime storage of their own tasks.

User-key programs of transactions defined with ISOLATE(NO) also have access to the user-key task-lifetime storage of other tasks defined with ISOLATE(NO).

See the description of the EXECKEY option on the PROGRAM definition for more information about task storage protection.

CICS

CICS obtains CICS-key storage for this transaction. Application programs executing in CICS key can both read and modify these storage areas. Application programs executing in user key can only read these storage areas.

Program resource definition changes

The meaning of the EXECKEY keyword of the program resource definition is changed.

EXECKEY(USER|CICS)

This specifies the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. For all except reentrant programs (that is, programs link-edited with the RENT attribute), EXECKEY also defines, in conjunction with the residency mode, into which of the DSAs CICS loads the program (see note 2).

USER

This specifies that CICS is to give control to the program in user key when it is invoked. CICS loads the program into one of the user-key shared DSAs—either the SDSA or the ESDSA, depending on the residency mode specified for the program (see note 2).

In a CICS region with storage protection only active, a user-key program has read and write access to all user-key storage, but read-only access to CICS-key storage.

In a storage protection **and** transaction isolation environment, a user-key program has read and write access to the user-key task-lifetime storage of its own task only, and to any shared DSA storage, if the transaction is defined with ISOLATE(YES).

However, if a transaction is defined with ISOLATE(NO) in a transaction isolation environment, its user-key programs also have read and write access to the user-key task-lifetime storage of other transactions that are defined with ISOLATE(NO).

User-key programs always have read-only access to CICS-key storage.

CICS

This specifies that CICS is to give control to the program in CICS key when it is invoked. CICS loads the program into one of the CICS-key DSAs—either the CDSA or the ECDSA, depending on the residency mode specified for the program (see note 2).

In a CICS region with storage protection active, a CICS-key program has read and write access to CICS-key and user-key storage of its own task and all other tasks, whether or not transaction isolation is active.

Notes:

1. First-level global user exit programs, task-related user exit programs, user-replaceable programs, and PLT programs always execute in CICS key, regardless of the EXECKEY definition.
2. If the program is link-edited with the RENT attribute, CICS loads the program into one of the read-only DSAs—either the RDSA or the ERDSA, depending on the residency mode specified for the program, regardless of the EXECKEY option. The read-only DSAs are allocated from read-only

storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

For more information about the different dynamic storage areas that CICS supports, see the *CICS/ESA System Definition Guide*.

Changes to the application programming interface

The CICS application programming interface (API) is extended. There are:

- Changes to the EXEC CICS ASSIGN command.
- Changes affecting the operation of the GETMAIN command.
- Some additions to the condition codes for the WAIT EVENT, WAIT EXTERNAL, and WAITCICS commands.
- A new transaction abend, AEYD, which occurs if CICS detects that an application program has asked CICS to write into an area to which the application does not have addressability. Note that CICS validates addresses ***before*** executing the command that could cause the storage violation. This means that in some cases where transactions would previously have caused an ASRA abend, they will now abend with AEYD.

EXEC CICS ASSIGN command

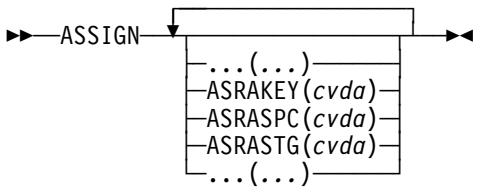
General-use programming interface

Three new option, ASRAKEY, ASRASPC, and ASRASTG are added to the EXEC CICS ASSIGN command. These are introduced to provide additional diagnostic information in the event of ASRA, ASRB, AICA, or AEYD abends.

Function

Request system values from outside the application program's local environment.

Syntax



Conditions:
INVREQ

ASSIGN gets values from outside the local environment of the application program.

ASSIGN options

ASRAKEY(cvda)

returns the execution key at the time of the last ASRA, ASRB, AICA, or AEYD, abend, if any. The CVDA values on the ASRAKEY option are as follows:

CICSEXECKEY
is returned if the task was executing in CICS-key at the time of the last ASRA, ASRB, AICA, or AEYD abend. Note that all programs execute in CICS key if CICS subsystem storage protection is not active.

USEREXECKEY
is returned if the task was executing in user-key at the time of the last ASRA, ASRB, AICA, or AEYD abend.

NONCICS is returned if the execution key at the time of the last abend was not one of the CICS keys; that is, not key 8 or key 9.

NOTAPPLIC is returned if there has not been an ASRA, ASRB, AICA, or AEYD abend.

ASRASPC(cvda)
returns the type of space in control at the time of the last ASRA, ASRB, AICA, or AEYD, abend, if any. The CVDA values on the ASRASPC option are:

SUBSPACE is returned if the task was executing in either its own subspace or the common subspace at the time of the last ASRA, ASRB, AICA, or AEYD abend.

BASESPACE is returned if the task was executing in the base space at the time of the last ASRA, ASRB, AICA, or AEYD abend. Note that all tasks execute in base space if transaction isolation is not active.

NOTAPPLIC is returned if there has not been an ASRA, ASRB, AICA, or AEYD abend.

ASRASTG(cvda)
returns the type of storage being addressed at the time of the last ASRA or AEYD, abend, if any. The CVDA values on the ASRASTG option are:

CICS is returned if the storage being addressed is CICS-key storage. This can be in one of the CICS dynamic storage areas (CDSA or ECDSA), or, in one of the read-only dynamic storage areas (RDSA or ERDSA) when CICS is running with the NOPROTECT option on the RENTPGM system initialization parameter or when storage protection is not active.

USER is returned if the storage being addressed is user-key storage in one of the user dynamic storage areas (UDSA or EUDSA).

READONLY is returned if the storage being addressed is read-only storage in one of the read-only dynamic storage areas (RDSA or ERDSA) when CICS is running with the PROTECT option on the RENTPGM system initialization parameter.

NOTAPPLIC is returned if:

- There is no ASRA or AEYD abend found for this task
- The affected storage in an abend is not managed by CICS
- The ASRA abend is not caused by an 0C4 abend
- An ASRB or AICA abend has occurred since the last ASRA or AEYD abend.

For ASSIGN command options introduced in support of other new function, see Chapter 34, “Changes to the application programming interface (API)” on page 567.

GETMAIN

There is no change to the syntax of the EXEC CICS GETMAIN command, but there are some changes to the operation of the command to support transaction isolation and the new shared dynamic storage areas (SDSA and ESDSA).

GETMAIN gets a main storage area of the size indicated by the FLENGTH option. (You can also use the LENGTH option, but this is supported for compatibility purposes and you are strongly recommended to use FLENGTH.) The address of the area is returned in the pointer reference supplied in the SET option.

CICS always allocates on double-word boundaries and rounds the requested length up to the nearest double-word multiple. Because there is no default initialization, you must use the INITIMG option if you require the storage to be initialized to a specific bit configuration.

CICS allocates storage from one of six different dynamic storage areas (DSAs):

- The CICS dynamic storage area (CDSA), below 16MB
- The user dynamic storage area (UDSA), below 16MB
- The shared dynamic storage area (SDSA), below 16MB
- The extended CICS dynamic storage area (ECDSA), above 16MB
- The extended user dynamic storage area (EUDSA), above 16MB
- The extended shared dynamic storage area (ESDSA), above 16MB.

Note: There are two other dynamic storage areas—the read-only DSA (RDSA) and the extended read-only DSA (ERDSA)—but you cannot GETMAIN storage from these DSAs.

CICS determines whether to obtain the requested storage above or below 16MB, from one of the CICS- or user-key DSAs, or from one of the shared DSAs, according to the following options:

- The FLENGTH option with BELOW also specified
- The FLENGTH option alone, in conjunction with the addressing mode of the requesting program
- The LENGTH option
- The SHARED option.

The above- or below-the-line allocation is summarized in the following table:

Options	Address mode	
	24-bit	31-bit
FLENGTH BELOW	A DSA	A DSA
FLENGTH alone	A DSA	An EDSA
LENGTH	A DSA	A DSA

CICS decides whether to allocate storage from a CICS-key or user-key DSA, or from a shared DSA, according to the following options:

- The USERDATAKEY option on the GETMAIN command.
- The CICSDATAKEY option on the GETMAIN command.
- The TASKDATAKEY option on the transaction resource definition under which the requesting program is running if the USERDATAKEY or CICSDATAKEY option is omitted.
- The SHARED option on the GETMAIN command.

The data-key option on the GETMAIN command overrides the TASKDATAKEY option on the transaction resource definition. The effect of the data-key options, without the SHARED option, is summarized in the following table:

Without SHARED option		
No data-key option	**USERDATAKEY specified**	**CICSDATAKEY specified**
Determined by TASKDATAKEY on transaction definition	User-key storage (from UDSA or EUDSA)	CICS-key storage (from CDSA or ECDSA)

The effect of the SHARED option is summarized in the following table:

With SHARED option		
No data-key option	**USERDATAKEY specified**	**CICSDATAKEY specified**
Determined by TASKDATAKEY on transaction definition	User-key storage (from SDSA or ESDSA)	CICS-key storage (from CDSA or ECDSA)

The storage that a task gets is available until it is released with a FREEMAIN command. For an area obtained without the SHARED option, only the task that acquired the storage may release it, and at task end CICS automatically releases such storage not already released. Note that any storage acquired with the SHARED option is accessible by all tasks, including those that are running with transaction isolation.

A SHARED area, on the other hand, is not released at task end and remains until explicitly freed; any task may issue the FREEMAIN. This means that you can use SHARED storage in task-to-task communication.

You cannot, however, use the storage obtained as a TIOA for subsequent terminal operations, because this could cause storage violations.

The following COBOL example shows how to get a 1024-byte area from user-key storage below 16MB (assuming that TASKDATAKEY(USER) is specified on the transaction resource definition), and initialize it to spaces:

```
EXEC CICS GETMAIN SET(PTR)
          FLENGTH(1024)
          BELOW
          INITIMG(BLANK)
```

You must define BLANK in your program as the character representing a space.

The following COBOL example shows how to get a 2048-byte area from CICS-key storage above 16MB (regardless of the TASKDATAKEY option specified on the transaction resource definition), and initialize it to spaces:

```
EXEC CICS GETMAIN SET(PTR)
          FLENGTH(2048)
          INITIMG(BLANK)
          CICSDATAKEY
```

Specifying CICSDATAKEY ensures that the requesting program obtains CICS-key storage from a CICS DSA, even if TASKDATAKEY(USER) is specified on the transaction resource definition.

New condition codes for WAIT EVENT, WAIT EXTERNAL and WAITCICS

There are changes to the WAIT EVENT, WAIT EXTERNAL, and WAITCICS API command condition codes. Storage for the timer-event control area on EXEC CICS WAIT EVENT and storage for event control blocks (ECBs) specified on EXEC CICS WAIT EXTERNAL and EXEC CICS WAITCICS commands must reside in shared storage if you have specified ISOLATE(YES).

These commands return an INVREQ response with a RESP2 value of 6 if:

- The timer-event control area specified on a WAIT EVENT is in user-key task-lifetime storage, and is inaccessible to another transaction. This condition can only occur if the storage for the timer-event control area is obtained other than by an EXEC CICS POST command, and is for posting as an ECB by some other task on completion of an event (RESP2=6).

  Note: CICS obtains storage for a timer-event control area in response to an EXEC CICS POST command (and which can be used in conjunction with the WAIT EVENT command) from a shared subpool in user-key storage. This ensures that timer-event control areas are in shared storage and, when referenced by a subsequent WAIT EVENT command, do not fail with an INVREQ.
- The event control blocks (ECBs) specified are in user-key task-lifetime storage, and are inaccessible to another transaction that is expected to post the ECBs (RESP2=6). (WAIT EXTERNAL or WAITCICS)
- The timer-event control area specified is in read-only storage (RESP2=6). (WAIT EVENT)
- The ECBs specified are in read-only storage (RESP2=6). (WAIT EXTERNAL or WAITCICS)

If CICS is executing with transaction isolation active and the transaction is defined with ISOLATE(YES), CICS checks that the timer-event control area or the ECBs are in shared storage.

If CICS is executing with or without transaction isolation, CICS checks that the timer-event control area or the ECBs are not in read-only storage. These checks are summarized in Table 2 on page 31.

Table 2. Location of event control blocks that cause an INVREQ (RESP2=6)			
RENTPGM option	**TRANISO option**	**ISOLATE option**	**ECB storage**
–	Yes	Yes	User-key, non-shared
PROTECT	–	–	Read-only storage

To help you identify potential problems with programs that issue WAIT EVENT, WAIT EXTERNAL, and WAITCICS commands, you can use the scanner and detector components of the IBM CICS Transaction Affinities Utility MVS/ESA. The scanner also detects programs that issue MVS POST macros. See Chapter 8, “The IBM CICS Transaction Affinities Utility MVS/ESA” on page 123 for brief details of this CICS utility.

Address validation of addresses passed to CICS

If you specify the system initialization parameter CMDPROT=YES, CICS validates all API commands issued by an application program to determine whether the program itself has write access to any referenced storage. This ensures that CICS does not perform storage violations on behalf of the application program, and modify storage that the application program does not own. If the application program itself does not have write access to the storage referenced by the CICS command, the transaction is abended with an AEYD abend, prior to the execution of the offending command.

For example, if a task executing in user key issues the following command:

```
EXEC CICS READ FILE(file-name)
     INTO(data-area)
```

and the task itself does not have write access to *data-area*, CICS abends the transaction.

This command protection prevents CICS overwriting storage to which the user task itself does not have write access.

The level of protection against bad addresses depends on the level of storage protection in the CICS environment. For example, if you are running CICS with CICS subsystem storage protection active, CICS protects all CICS-key storage in the CICS dynamic storage areas (CDSA and ECDSA). The various levels of protection are shown in Table 3.

Table 3 (Page 1 of 2). Levels of protection provided by CICS validation of application-supplied addresses		
Environment	**Execution key of affected programs**	**Types of storage referenced by applications that cause AEYD abends**
Read-only storage (RENTPGM=PROTECT)	CICS-key and user-key	CICS key 0 read-only storage (RDSA and ERDSA).
Subsystem storage protection (STGPROT=YES)	User-key	All CICS-key storage (CDSA and ECDSA)
Transaction isolation (TRANISO=YES)	User-key and ISOLATE(YES)	Task-lifetime storage of all other transactions

Table 3 (Page 2 of 2). Levels of protection provided by CICS validation of application-supplied addresses

Environment	Execution key of affected programs	Types of storage referenced by applications that cause AEYD abends
Transaction isolation (TRANISO=YES)	User-key and ISOLATE(NO)	Task-lifetime storage of all **except** other user key and ISOLATE(NO) transactions
Base CICS (all storage is CICS key 8 storage) (RENTPGM=NOPROTECT; STGPROT=NO; and TRANISO=NO)	CICS-key and user-key	MVS storage only

This protection applies only to the CICS application programming and system programming interfaces. Other programming interfaces such as DL/1, DB2, and other resource managers are not similarly protected.

The commands that can give rise to this condition are those that, in general terms, provide output fields ("receiver" fields). These are described in the CICS programming reference manuals as having parameter values of the type *data-area*. See the *CICS/ESA System Programming Reference* manual for a description of input and output parameter values.

Changes to the system programming interface

The CICS system programming interface (SPI) is extended with additions and changes to INQUIRE and SET commands:

- EXEC CICS INQUIRE STORAGE
- EXEC CICS INQUIRE SYSTEM
- EXEC CICS SET SYSTEM
- EXEC CICS INQUIRE TASK
- EXEC CICS INQUIRE TRANSACTION

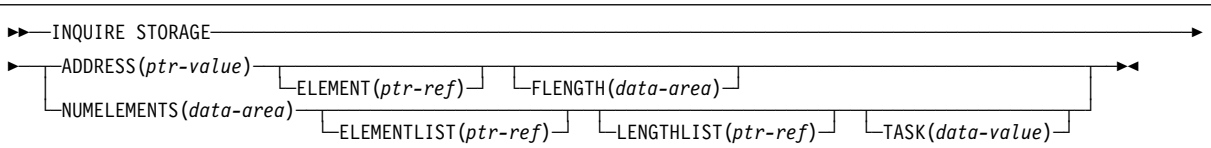
EXEC CICS INQUIRE STORAGE command

This is a new command that allows you to retrieve information about elements of task-lifetime storage.

Function

Retrieve information about elements of task-lifetime storage.

Syntax



Conditions:
NOTAUTH, TASKIDERR

INQUIRE STORAGE options

There are two forms for the INQUIRE STORAGE command, each of which has a number of optional parameters:

1. In format 1 you specify the ADDRESS parameter
2. In format 2 you specify the NUMELEMENTS parameter.

Options for format 1

ADDRESS(ptr-value)
specifies a pointer reference to an element of task-lifetime storage, of which you want CICS to return the length.

The address you pass can reference any byte within the element of storage—it does not have to address the start of the area. If the storage addressed by the pointer is a valid element of task-lifetime storage, CICS returns information in the ELEMENT and FLENGTH operands.

ELEMENT(ptr-ref)
returns the start address of the element of task-lifetime storage referenced by the ADDRESS operand.

If the ADDRESS pointer reference you pass to CICS addresses the start of the element of task-lifetime storage, the address CICS returns on the ELEMENT operand is the same.

If the ADDRESS pointer reference you pass to CICS does not address any part of any element of task-lifetime storage, CICS returnsX'FF000000'.

FLENGTH(data-area)
returns, as a full-word binary value, the length of the element of task-lifetime storage referenced by the ADDRESS operand. The length CICS returns is of the user portion of the storage element, excluding the storage leading and trailing storage check zones.

Options for format 2

ELEMENTLIST(ptr-ref)
returns the address of an area of storage that contains a list of addresses of all the elements of task-lifetime storage, either for:

The task issuing the INQUIRE STORAGE request, or
The task specified on the TASK parameter.

The length of the list, and the number of addresses in it, are returned in the LENGTHLIST and NUMELEMENTS operands.

LENGTHLIST(ptr-ref)
returns the address of an area of storage that contains a list of the lengths of the elements of task-lifetime storage. The lengths correspond to the areas of storage whose addresses are returned in the list referenced by the ELEMENTLIST operand.

NUMELEMENTS(data-area)
returns, as a full-word binary value, the number of elements of storage referenced by the list of element addresses.

TASK(data-value)
specifies, as a 4-byte packed decimal value, the task number for which you want CICS to return details of the task-lifetime storage. If you omit the task number, CICS assumes the inquiry is for the task issuing the INQUIRE STORAGE command.

INQUIRE STORAGE conditions

Condition	*RESP2*	*Meaning*
NOTAUTH	100	The use of this command is not authorized.
TASKIDERR	1	Task number does not exist.
TASKIDERR	2	The task number is a system task, not a user task.

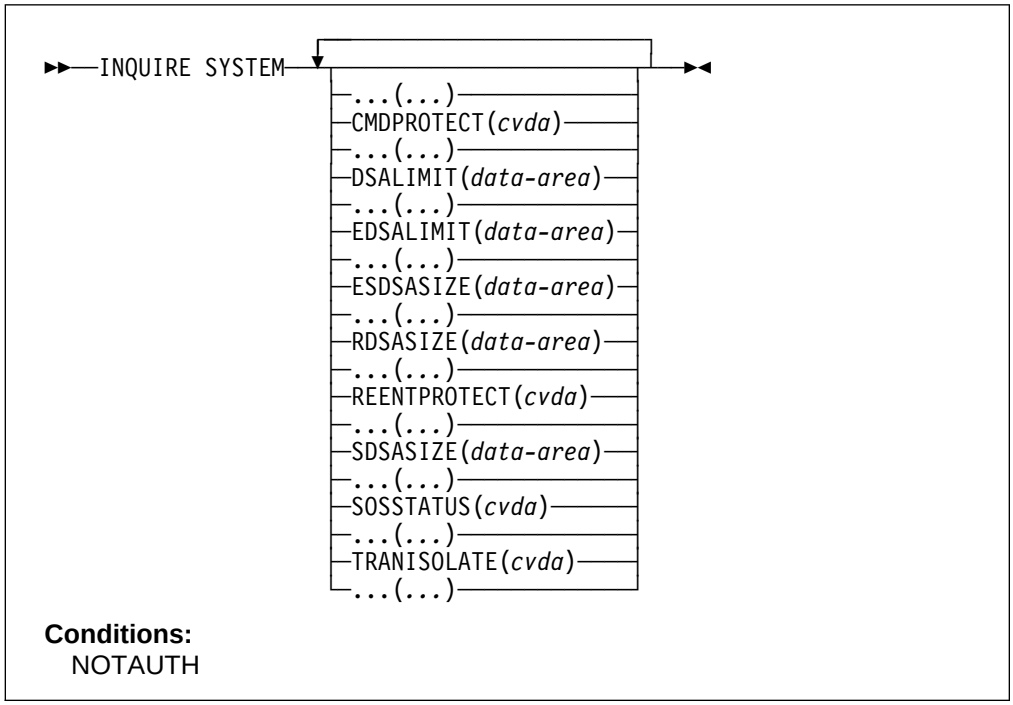
EXEC CICS INQUIRE SYSTEM command

The five storage cushion size options are removed in CICS/ESA 4.1. These obsolete keywords are: CSCS, ECSCS, ERSCS, EUSCS, and USCS.

Some new options are added, and these are shown in the command syntax.

Function
Retrieve information about a specified system definition.

Syntax



Conditions:
NOTAUTH

INQUIRE SYSTEM options

CMDPROTECT(cvda)
returns a CVDA to indicate whether command protection, which validates start addresses passed on CICS commands, is active or not (that is, whether the CMDPROT system initialization parameter specifies YES or NO). The CVDA values are:

CMDPROT Command protection is active. CICS checks to ensure the task itself has write access to the storage referenced on the command before writing to the storage on the task's behalf.

NOCMDPROT Command protection is not active. CICS does not check whether the task itself has write access to the storage referenced on the command before writing to storage on the task's behalf.

DSALIMIT(data-area)
returns a full word binary field indicating the maximum amount of storage, as a total number of bytes, within which CICS can dynamically allocate storage for the four individual DSAs that reside below 16MB.

EDSALIMIT(data-area)
returns a full word binary field indicating the maximum amount of storage, as a total number of bytes, within which CICS can dynamically allocate storage for the four individual DSAs that reside above 16MB.

ESDSASIZE(data-area)
returns a full word binary field indicating the current size of the extended shared dynamic storage area (ESDSA). The size of this storage area is calculated and managed by CICS automatically, within the overall limits specified for all the DSAs that reside above 16MB.

RDSASIZE(data-area)
returns a full word binary field indicating the current size of the read-only dynamic storage area (RDSA). The size of this storage area is calculated and managed by CICS automatically, within the overall limits specified for all the DSAs that reside below 16MB.

REENTPROTECT(cvda)
returns a CVDA to indicate whether read-only storage is in use for reentrant programs (that is, whether the RENTPGM system initialization parameter specifies PROTECT or NOPROTECT). CVDA values are:

REENTPROT
CICS allocates storage for the read-only DSAs (RDSA and ERDSA) from key-0, non-fetch protected, storage. CICS loads reentrant programs into this storage, which are protected by residing in read-only storage.

NOREENTPROT
CICS allocates storage for the RDSA and ERDSA from CICS-key storage. Reentrant programs do not have the protection of residing in read-only storage, and can be modified by programs executing in CICS key.

SDSASIZE(data-area)
returns a full word binary field indicating the current size of the shared dynamic storage area (SDSA). The size of this storage area is calculated and managed by CICS automatically, within the overall limits specified for all the DSAs that reside below 16MB.

SOSSTATUS
returns a CVDA to indicate whether CICS is short on storage. CVDA values are:

NOTSOS CICS is not short on storage in any of the dynamic storage areas.

SOS CICS is short on storage in at least one of the dynamic storage areas above and below 16MB.

SOSABOVE
CICS is short on storage in at least one of the dynamic storage areas above 16MB.

SOSBELOW
CICS is short on storage in at least one of the dynamic storage areas below 16MB.

TRANISOLATE(cvda)
returns a CVDA value indicating the status of transaction isolation. CVDA values are:

ACTIVE
Transaction isolation is active in the CICS region.

INACTIVE
CICS is running without transaction isolation, either because the support is not available, or because it was not requested at CICS initialization.

Note: The descriptions of the existing xDSASIZE options are also changed. They return the current size of the dynamic storage area, the size being calculated and managed by CICS automatically, within the overall limits specified for all the DSAs that reside below 16MB.

The five storage cushion size options (keywords CSCS, ECSCS, ERSCS, EUSCS, and USCS) are obsolete in CICS/ESA 4.1, but the other options of the INQUIRE SYSTEM command remain unchanged from CICS/ESA 3.3.

Note that although the cushion size options are no longer supported, and are removed from the CICS translator language tables, they are accepted at run time on existing applications to maintain object compatibility. At run time, CICS ignores any xxSCS options and they have no effect, but the translator treats them as errors.

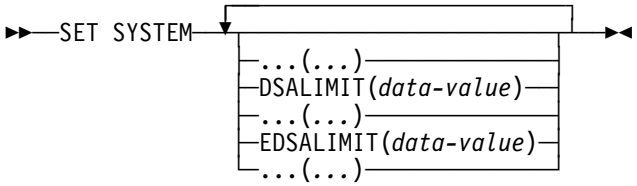
EXEC CICS SET SYSTEM command

The five storage cushion size options are removed in CICS/ESA 4.1. The obsolete keywords are: CSCS, ECSCS, ERSCS, EUSCS, and USCS. Two new options are added for setting the DSA overall size limits. These are shown in the command syntax below.

Function

Change the value of system attributes.

Syntax



Conditions:
INVREQ, NOTAUTH, NOSTG

SET SYSTEM options

DSALIMIT(data-value)

specifies, as a full-word binary value, the maximum amount of storage, as a total number of bytes, within which CICS can dynamically allocate storage for the four individual DSAs that reside below 16MB.

If DSALIMIT specifies a value lower than the current limit, CICS may not be able to implement the new limit immediately, but will attempt to do so over time as dynamic storage is freed in the individual DSAs.

Changes to the DSA limit are cataloged, and therefore are preserved across a CICS restart. Note, however, that the change is overridden by a DSALIM system initialization parameter that is specified in SYSIN or in a PARM statement in the startup JCL.

EDSALIMIT(data-value)

specifies, as a full-word binary value, the maximum amount of storage, as a total number of bytes, within which CICS can dynamically allocate storage for the four individual DSAs that reside above 16MB.

If EDSALIMIT specifies a value lower than the current limit, CICS may not be able to implement the new limit immediately, but will attempt to do so over time as dynamic storage is freed in the individual DSAs.

Changes to the EDSA limit are cataloged, and therefore are preserved across a CICS restart. Note, however, that the change is overridden by an EDSALIM system initialization parameter that is specified in SYSIN or in a PARM statement in the startup JCL.

The CSCS, ECSCS, USCS, EUSCS, and ERSCS options are no longer supported. To maintain object compatibility, they are accepted by CICS at run time, but ignored. The options are removed from the translator language tables, and cause a translator error if present in source programs at translate-time.

SET SYSTEM conditions

The RESP2 values on the INVREQ condition are extended:

Condition	*RESP2*	*Meaning*
INVREQ	20	DSALIMIT is not in the range 2MB to 16MB.
INVREQ	21	EDSALIMIT is not in the range 10MB to 2GB.
INVREQ	22	There is insufficient MVS storage to allocate the specified DSALIMIT.

Condition	*RESP2*	*Meaning*
INVREQ	23	There is insufficient MVS storage to allocate the specified EDSALIMIT.

Current RESP2 values for cushion sizes out of range are removed.

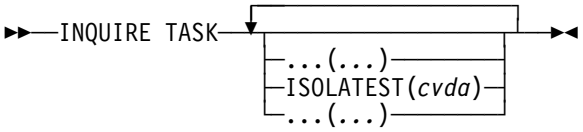
EXEC CICS INQUIRE TASK and INQUIRE TRANSACTION

INQUIRE TASK and INQUIRE TRANSACTION are extended to include the ISOLATEST option, which indicates whether the transaction is defined, or the task executing, with transaction isolation.

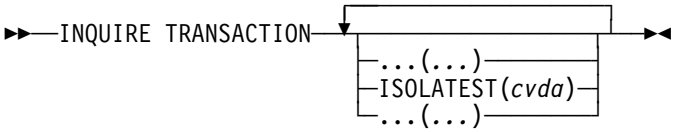
Function

Returns information about a named transaction or task.

Syntax



Conditions:
NOTAUTH, TASKIDERR



Conditions:
NOTAUTH, TRANSIDERR, ILLOGIC

INQUIRE TASK and INQUIRE TRANSACTION options

ISOLATEST(cvda)

returns a CVDA value identifying whether the user-key task-lifetime storage is isolated from the user-key programs of other transactions. CVDA values are:

ISOLATE

The user-key task-lifetime storage is accessible only by the user-key programs of its own task, and also CICS-key programs of other tasks. The user-key task-lifetime storage is isolated from all the user-key programs of all other tasks.

Also, the converse is true: the user key programs of this task cannot access user-key task-lifetime storage belonging to other tasks.

NOISOLATE

The task's user-key task-lifetime storage is accessible by its own programs, and also by user-key programs of other transactions defined with the ISOLATE(NO) option, and CICS-key programs of other tasks.

Also, the user key programs of this task cannot access user-key task-lifetime storage belonging to tasks that are defined with ISOLATE(YES).

Notes:

1. Any storage obtained by EXEC CICS GETMAIN SHARED is not protected.
2. Task-lifetime storage is not protected from EXECKEY(CICS) programs.
3. The value of ISOLATEST is taken from the transaction definition, and does not take into account whether transaction isolation is enabled. To determine whether the task does, in fact, have its user-key task-lifetime storage isolated from other transactions, it is necessary to issue an EXEC CICS INQUIRE SYSTEM TRANISOLATE command. This returns an indication of whether transaction isolation is enabled.

End of General-use programming interface

Changes to global user exits

Product-sensitive programming interface

There are changes to the data passed to two global user exits, XPCTA and XPCHAIR.

Program control exits, XPCTA and XPCHAIR

Global user exit XPCHAIR is invoked immediately before a HANDLE ABEND LABEL routine is given control. Global user exit XPCTA is invoked immediately after an abend and before the environment is modified. In both cases the exits are invoked after CICS has created the task abend control block (TACB).

XPCTA allows passing of control to an address, and in an AMODE and execution key specified by the exit program. This interface is enhanced so that the exit program can also specify the subspace that control is passed in. The sample XPCTA exit program (DFH$PCTA) can be extended for transaction isolation.

Global user exits XPCTA and XPCHAIR both receive the address of the TACB as an exit-specific parameter. The TACB is extended to provide details of the subspace and access registers current at the time of the last abend (if relevant to that abend code). You can use the DFHTACB TYPE=DSECT macro to map the TACB control block.

See the *CICS/ESA Customization Guide* for more information about these global user exits.

Changes to the exit programming interface

There are four new function calls provided on the exit programming interface to the storage manager domain. These new functions, and the macro calls that support them, are as follows:

Function	Storage manager macro call
SWITCH_SUBSPACE	DFHSMSRX
INQUIRE_SHORT_ON_STORAGE	DFHSMSRX
INQUIRE_ELEMENT_LENGTH	DFHSMMCX
INQUIRE_TASK_STORAGE	DFHSMMCX

The SWITCH_SUBSPACE function

The SWITCH_SUBSPACE call's purpose is to cause CICS to switch from a subspace to base space, if the task is not already executing in the base space. If the task is already in the base space, storage manager ignores the call.

This function can be used by global user exit programs that receive control in subspace and for some reason need to switch into basespace.

The syntax of the call is shown as follows:

SWITCH_SUBSPACE

```
DFHSMSRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(SWITCH_SUBSPACE),
       SPACE(BASESPACE),]
       [OUT,
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

SPACE(BASESPACE)

Issue a switch to the basespace if the task issuing the call is currently executing within a subspace. This enables the task to read and write to another task's user-key task-lifetime storage.

RESPONSE and REASON values for SWITCH_SUBSPACE:

RESPONSE	*REASON*
OK	None
DISASTER	None
KERNERROR	None

The INQUIRE_SHORT_ON_STORAGE function

The INQUIRE_SHORT_ON_STORAGE call's purpose is to enable the caller to determine whether CICS is short on storage either above or below 16MB.

The syntax of the call is shown as follows:

INQUIRE_SHORT_ON_STORAGE

```
DFHSMSRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_SHORT_ON_STORAGE),]
       [OUT,
       SOS_BELOW_THE_LINE(YES|NO),
       SOS_ABOVE_THE_LINE(YES|NO),
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

SOS_BELOW_THE_LINE(YES|NO),

returns YES if CICS is currently short-on-storage in any of the DSAs below 16MB, and NO if not.

SOS_ABOVE_THE_LINE(YES|NO),
returns YES if CICS is currently short-on-storage in any of the DSAs above 16MB, and NO if not.

RESPONSE and REASON values for INQUIRE_SHORT_ON_STORAGE:

RESPONSE	*REASON*
OK	None
DISASTER	None
KERNERROR	None

The INQUIRE_ELEMENT_LENGTH function

The INQUIRE_ELEMENT_LENGTH call's purpose is to enable the caller to pass the address of any part of an element of task-lifetime storage, and to obtain from CICS the start address and the length of the storage element that contains the passed address.

The syntax of the call is shown as follows:

INQUIRE_ELEMENT_LENGTH

```
DFHSMMCX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION (INQUIRE_ELEMENT_LENGTH),
       ADDRESS (name4 | (Rn) | *),]
       [OUT,
       ELEMENT_ADDRESS(name4 | (Rn) | *),
       ELEMENT_LENGTH(name4 | (Rn) | *),
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

ADDRESS(name4 | (Rn) | *)
specifies an address that lies within an element of task-lifetime storage of the current task.

CICS accepts addresses that reference the leading or trailing check zones as being valid addresses for the element of storage you are inquiring upon.

ELEMENT_ADDRESS(name4 | (Rn) | *)
returns the start address of the element of task-lifetime storage referenced by the ADDRESS parameter. The start address returned does **not** include the leading check zone.

ELEMENT_LENGTH(name4 | (Rn) | *)
returns the length of the element of task-lifetime storage referenced by the ADDRESS parameter. The length returned does **not** include the leading or trailing check zones.

RESPONSE and REASON values for INQUIRE_ELEMENT_LENGTH:

RESPONSE	*REASON*
OK	None
EXCEPTION	INVALID_ADDRESS

DISASTER	None
INVALID	None
KERNERROR	None
PURGED	None

The INQUIRE_TASK_STORAGE function

The INQUIRE_TASK_STORAGE call's purpose is to enable the caller to request details of all elements of task-lifetime storage belonging to a task. You can specify the transaction number of the task explicitly on the call, or let it default to the current task.

The syntax of the call is shown as follows:

INQUIRE_TASK_STORAGE

```
DFHSMMCX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION (INQUIRE_TASK_STORAGE),
       [TRANSACTION_NUMBER(name4 | (Rn) | *),]
       ELEMENT_BUFFER(name4 | (Rn) | *),
       LENGTH_BUFFER(name4 | (Rn) | *),]
       [OUT,
       NUMBER_OF_ELEMENTS(name4 | (Rn) | *),
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

ELEMENT_BUFFER(name4 | (Rn) | *)
specifies the address of a buffer into which CICS returns a list of start addresses of all the elements of task-lifetime storage belonging to either the specified task or, by default, the current task.

The start addresses returned do **not** include the leading check zone.

LENGTH_BUFFER(name4 | (Rn) | *)
specifies the address of a buffer into which CICS returns a list of the lengths of the elements of task-lifetime storage belonging to either the specified task or, by default, the current task. The lengths returned do **not** include the leading or trailing check zones.

NUMBER_OF_ELEMENTS(name4 | (Rn) | *)
returns the number of entries in each of the two buffers, ELEMENT_BUFFER and LENGTH_BUFFER, as a full-word binary value.

TRANSACTION_NUMBER(name4 | (Rn) | *)
specifies, as a 4 byte packed decimal value, the transaction number of the task to whom the storage belongs.

If you omit the transaction (task) number, CICS assumes the current task.

RESPONSE and REASON values for INQUIRE_TASK_STORAGE:

RESPONSE	*REASON*
OK	None
EXCEPTION	INSUFFICIENT_STORAGE

	NO_TRANSACTION_ENVIRONMENT
DISASTER	None
INVALID	None
KERNERROR	None
PURGED	None

Changes to user replaceable modules

There are changes to the user-replaceable program error program, DFHPEP.

Program error program (DFHPEP)

The communications area (COMMAREA) passed to DFHPEP is extended to include information from the task abend control block (TACB) for errors that relate to transaction isolation.

The COMMAREA, which is mapped by the DFHPCOM DSECT, includes some new EQUATE values for the storage areas affected in ASRA abends. The new EQUATES are as follows:

```
*
PEP_COM_STORAGE_HIT           DS     X         storage type referenced
*                                              in abend (ASRA only)
 .
PEP_COM_RDSA_HIT              EQU    4         RDSA hit
PEP_COM_EUDSA_HIT             EQU    5         EUDSA hit
PEP_COM_UDSA_HIT              EQU    6         UDSA hit
```

The COMMAREA also includes a new field, with associated EQUATES, to indicate whether the abending task was executing in the base space or a subspace. The details of this, mapped by DFHPCOM, are as follows:

```
*
PEP_COM_SPACE                 DS     X
*
PEP_COM_NOSPACE               EQU     0
PEP_COM_SUBSPACE              EQU    10
PEP_COM_BASESPACE             EQU    11
*
```

For ASRA, ASRB, AICA, and AEYD abends, PEP_COM_SPACE is always PEP_COM_BASESPACE if transaction isolation is not active. For all other abends, PEP_COM_SPACE is always PEP_COM_NOSPACE.

See the *CICS/ESA Customization Guide* for more information about the DFHPEP user replaceable module.

End of Product-sensitive programming interface

Changes to CICS-supplied transactions

Most CICS-supplied transactions are defined as TASKDATAKEY(CICS) and ISOLATE(YES), which ensures that they execute in CICS key, and that any user-key storage they explicitly obtain is isolated from any user-key programs. This means their storage is protected from application overwrites.

The exceptions to this are the mirror transactions, CSMI, CPMI, CVMI, CSM1, CSM2, CSM3, and CSM5 used in function shipping and distributed program link

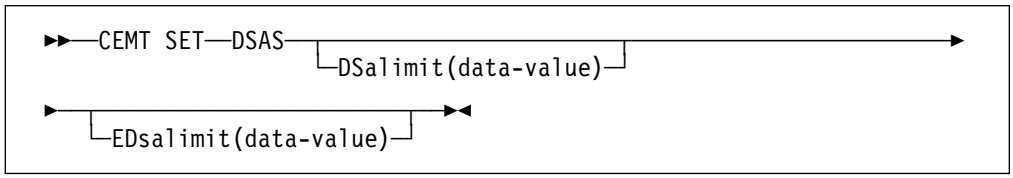
(DPL). These are defined as TASKDATAKEY(USER) and ISOLATE(YES), ensuring that any user-key storage they acquire is isolated from all user-key programs other than those executing under the mirrors.

CEMT INQUIRE and SET DSAS

These new commands provide a subset of the options that are also provided on the INQUIRE SYSTEM command.

The commands return information about all the individual DSAs in addition to the overall DSA storage limits (DSALIMIT and EDSALIMIT). The syntax of the commands, and a sample of the screen that is returned on the INQUIRE command, are shown below.

Command syntax



For details of the parameters on these CEMT INQUIRE and SET DSAS commands, see the descriptions given under the CEMT INQUIRE SYSTEM command.

```
INQ DSAS
STATUS:  RESULTS - OVERTYPE TO MODIFY

  SOSstatus(NOTSOS)
  Dsalimit( 05242880 )
   Cdsasize( 00524288 )
   Rdsasize( 00524288 )
   Sdsasize( 00000000 )
   Udsasize( 00000000 )

  EDsalimit( 0020971520 )
   ECdsasize( 0002097152 )
   ERdsasize( 0004194304 )
   ESdsasize( 0001048576 )
   EUdsasize( 0001048576 )









                                        SYSID=HTA1 APPLID=CICSHTA1
  RESPONSE: NORMAL                    TIME:  09.37.27  DATE: 09.30.93
PF 1 HELP       3 END                 7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

Figure 4. Example screen output from the CEMT INQUIRE DSAS command

CEMT INQUIRE and SET SYSTEM

(These commands are described more fully, with changes relating to other functions, in Chapter 36, “Changes to the master terminal transaction (CEMT)” on page 605.)

The CEMT INQUIRE and SET SYSTEM commands are changed in line with the CICS SPI changes.

The CEMT INQUIRE SYSTEM command returns the DSALIMIT and EDSALIMIT values. It also returns the individual DSA sizes currently set by CICS within the overall storage limits.

The CEMT SET SYSTEM command allows you to change the DSALIMIT and EDSALIMIT values.

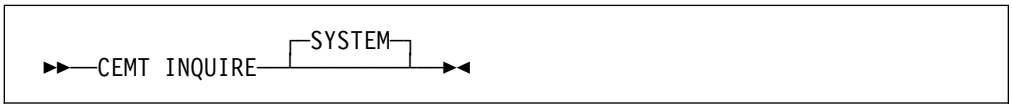
You cannot modify the status of:

- Transaction isolation while CICS is running (set by the TRANISO system initialization parameter)
- The read-only storage option for reentrant programs (set by the RENTPGM system initialization parameter)
- The command protection option (set by the CMDPROT system initialization parameter)

To change any of these, you must shut down CICS and restart.

The storage cushion options are removed from both the CEMT INQUIRE and SET commands.

Command syntax



Values displayed

CMdprotect(value)

displays whether command protection, which validates start addresses passed on CICS commands, is active or not (that is, whether the CMDPROT system initialization parameter specifies YES or NO). The values returned are:

CMDPROT Command protection is active. CICS checks to ensure the task itself has write access to the storage referenced on the command before writing to the storage on the task's behalf.

NOCMDPROT Command protection is not active. CICS does not check whether the task itself has write access to the storage referenced on the command before writing to storage on the task's behalf.

DSalimit(value)

displays the maximum amount of storage, as a total number of bytes, within which CICS can dynamically allocate storage for the four individual DSAs that reside below 16MB.

EDsalimit(value)
displays the maximum amount of storage, as a total number of bytes, within which CICS can dynamically allocate storage for the four individual DSAs that reside above 16MB.

ESdasize(value)
displays the current size of the extended shared dynamic storage area (ESDSA). The size of this storage area is calculated and managed by CICS automatically, within the overall limits specified for all the DSAs that reside above 16MB.

RDsasize(cvda)
displays the current size of the read-only dynamic storage area (RDSA). The size of this storage area is calculated and managed by CICS automatically, within the overall limits specified for all the DSAs that reside below 16MB.

REEntprotect(value)
displays whether read-only storage is in use for reentrant programs (that is, whether the RENTPGM system initialization parameter specifies PROTECT or NOPROTECT). The values displayed are:

REENTPROT CICS allocates storage for the read-only DSAs (RDSA and ERDSA) from key-0, non-fetch protected, storage. CICS loads reentrant programs into this storage, which are protected by residing in read-only storage.

NOREENTPROT CICS allocates storage for the RDSA and ERDSA from CICS-key storage. Reentrant programs do not have the protection of residing in read-only storage, and can be modified by programs executing in CICS key.

SDasize(value)
displays the current size of the shared dynamic storage area (SDSA). The size of this storage area is calculated and managed by CICS automatically, within the overall limits specified for all the DSAs that reside below 16MB.

SOSAbove(value)
displays whether CICS is short on storage. The values displayed are:

NOTSOS CICS is not short on storage in any of the dynamic storage areas.

SOS CICS is short on storage in at least one of the dynamic storage areas above and below 16MB.

SOSABOVE
CICS is short on storage in at least one of the dynamic storage areas above 16MB.

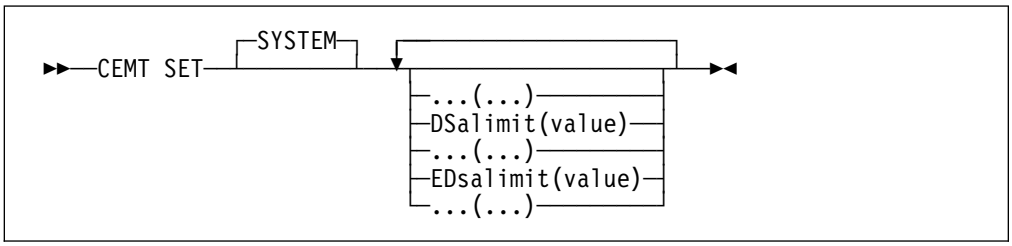
SOSBELOW
CICS is short on storage in at least one of the dynamic storage areas below 16MB.

TRanisolate(value)
displays the status of transaction isolation. The values returned are:

ACTIVE
Transaction isolation is active in the CICS region.

INACTIVE
CICS is running without transaction isolation, either because the support is not available, or because it was not requested at CICS initialization.

Note: The descriptions of the existing xDSASIZE options are also changed. They return the current size of the dynamic storage area, the size being calculated and managed by CICS automatically, within the overall limits specified for all the DSAs that reside below 16MB.



Command options

DSalimit(value)
specifies the maximum amount of storage, as a total number of bytes, within which CICS can dynamically allocate storage for the four individual DSAs that reside below 16MB.

If DSALIMIT specifies a value lower than the current limit, CICS may not be able to implement the new limit immediately, but will attempt to do so over time as dynamic storage is freed in the individual DSAs.

Changes to the DSA limit are cataloged, and therefore are preserved across a CICS restart. Note, however, that the change is overridden by a DSALIM system initialization parameter that is specified in SYSIN or in a PARM statement in the startup JCL.

EDsalimit(value)
specifies the maximum amount of storage, as a total number of bytes, within which CICS can dynamically allocate storage for the four individual DSAs that reside above 16MB.

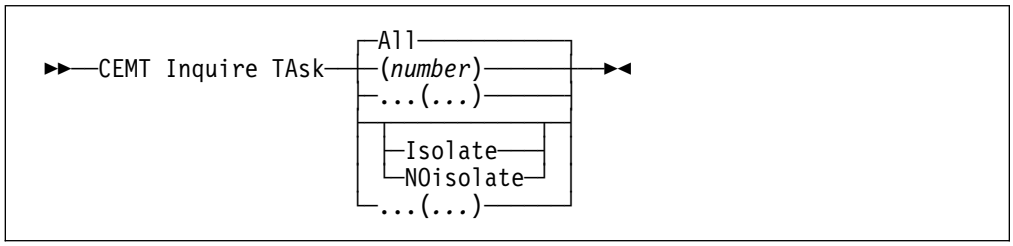
If EDSALIMIT specifies a value lower than the current limit, CICS may not be able to implement the new limit immediately, but will attempt to do so over time as dynamic storage is freed in the individual DSAs.

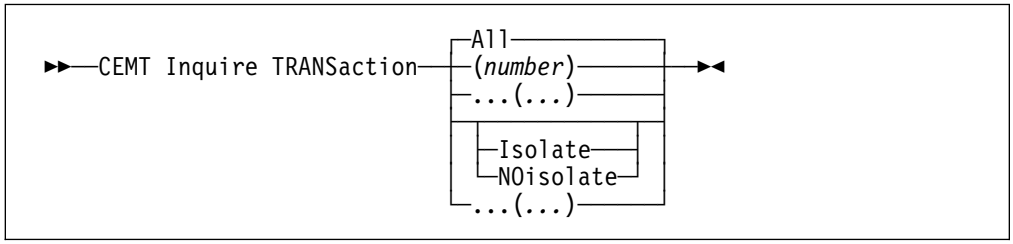
Changes to the EDSA limit are cataloged, and therefore are preserved across a CICS restart. Note, however, that the change is overridden by an EDSALIM system initialization parameter that is specified in SYSIN or in a PARM statement in the startup JCL.

CEMT INQUIRE TASK and CEMT INQUIRE TRANSACTION commands

The INQUIRE TASK and INQUIRE TRANSACTION commands are extended to include the ISOLATE and NOISOLATE options, which indicate whether the transaction is defined, or the task executing, with transaction isolation.

Command syntax





Isolate|Noisolate
displays whether the user-key task-lifetime storage is isolated from the user-key programs of other transactions. The values returned are:

ISOLATE
The user-key task-lifetime storage is accessible only by the user-key programs of its own task, and also CICS-key programs of other tasks. The user-key task-lifetime storage is isolated from all the user-key programs of all other tasks.

Also, the converse is true: the user key programs of this task cannot access user-key task-lifetime storage belonging to other tasks.

NOISOLATE
The task's user-key task-lifetime storage is accessible by its own programs, and also by user-key programs of other transactions defined with the ISOLATE(NO) option, and CICS-key programs of other tasks.

Also, the user key programs of this task cannot access user-key task-lifetime storage belonging to tasks that are defined with ISOLATE(YES).

Changes to monitoring and statistics

There are changes to CICS monitoring and statistics data as a result of the transaction isolation changes.

Monitoring

CICS monitoring is extended within the storage manager domain and the new program manager domain to monitor storage usage from within all the DSAs.

Statistics

New statistics and extensions of existing statistics are provided for storage manager.

See Chapter 18, "Monitoring and statistics" on page 295 for details of the monitoring and statistics changes.

Problem determination

CICS transaction and system dumps are changed for transaction isolation. These changes are mainly in the system data presented.

At the time of an attempted application overwrite, CICS ensures that the name of the abending program, and the area that it tried to overwrite, is made available to the program error program (DFHPEP) and to program control global user exit programs. (See “Program error program (DFHPEP)” on page 43 for more information.)

When an application program passes to CICS the address of some storage for CICS to store data into, the address is validated to check that the application itself could have stored data in it. If this check fails, CICS abnormally terminates the transaction with an AEYD abend code. Note that CICS checks only the start address of storage areas; that is, the first byte, not the entire storage area.

Messages and codes

CICS storage manager messages are extended as part of transaction isolation. The following messages are introduced in CICS/ESA 4.1 in support of the transaction isolation facility:

DFHSM0122
DFHSM0123
DFHSM0124
DFHSM0125
DFHSM0126
DFHSM0127
DFHSM0128
DFHSM0129
DFHSM0130
DFHSM0131
DFHSM0132
DFHSM0133
DFHSM0134

The following messages are obsolete in CICS/ESA 4.1:

DFHSM0112
DFHSM0116
DFHSM0117
DFHSM0118
DFHSM0121

Message DFHSR0622 is changed in CICS/ESA 4.1.

For details of all the new, changed, and obsolete messages in CICS/ESA 4.1, see the *CICS/ESA Migration Guide*.

Chapter 3. VTAM persistent sessions

This chapter describes CICS/ESA 4.1's use of, and the benefits it derives from, VTAM persistent sessions. It covers the following topics:

- Overview
- Benefits of VTAM persistent sessions
- Requirements for CICS/ESA 4.1 to use VTAM persistent sessions
- Changes to CICS externals
- Using FEPI with VTAM persistent sessions.

Overview

Persistent session support improves the availability of CICS. It benefits from VTAM 3.4.1 persistent LU–LU session improvements to provide restart-in-place of a failed CICS without rebinding.

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 rebind them.

You can change the interval 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 in an emergency restart.

If CICS is terminated through CEMT PERFORM SHUTDOWN IMMEDIATE, or if CICS fails, its sessions are placed 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 VTAM ACB.

Subsequent processing is LU dependent: cleanup and recovery for non-LU6 persistent sessions are similar to that for non-LU6 backup sessions under XRF. Cleanup and recovery for LU6.2 persistent sessions maintain the bound session when possible but there are cases where it is necessary to unbind and rebind 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 session support and fails, the user perception is that CICS is "hanging": what is on the screen at the time of the failure remains until persistent session recovery is complete. After a successful CICS emergency restart, the recovery options defined for the terminals or sessions take effect.

The recovery options are specified on the RECOVOPTION parameter of the TYPETERM resource definition. If you specify SYSDEFAULT as the value for RECOVOPTION, the terminal user can clear the screen, re-signon if they were signed on at the time of failure, and continue to enter CICS transids. If you specify MESSAGE as the RECOVNOTIFY attribute of the TYPETERM resource definition, the user is notified of the successful recovery, after which they can continue as in the SYSDEFAULT case.

Note: CICS does not maintain a user's signon in the event of a failure with persistent session support. If a user is signed on at the time of a CICS failure, they need to signon again after their session is recovered, to re-establish their security environment.

Multi-node persistent sessions

APAR for multi-node persistent sessions

For multi-node persistent sessions support, you need the relevant PTF for APAR PQ01573.

If you are running CICS with VTAM 4.4 or later and are using multi-node persistent sessions (MNPS) support, sessions can be retained across a VTAM failure. After VTAM has been restarted, sessions are restored when the ACB is reopened (either automatically by the COVR transaction or by a CEMT, or EXEC CICS, SET VTAM OPEN command). To ensure that CICS retains its sessions, and restores them when the ACB is reopened, code PSTYPE=MNPS as a system initialization parameter.

APPC synclevel 2 sessions are not restored. They are unbound when the ACB is opened.

When the VTAM failure occurs and the TPEND failure exit is driven, the autoinstalled terminals that are normally deleted at this point are retained by CICS. If the session is not restored and the terminal is not reused within the AIRDELAY interval, CICS deletes the TCTTE when the AIRDELAY interval expires after the ACB is re-opened successfully.

Single-node persistent sessions

If you are not using MNPS, you don't need a PSTYPE system initialization parameter and sessions are not retained or recovered after a VTAM abend and subsequent opening of the VTAM ACB (CEMT SET VTAM OPEN).

Unbinding sessions

Sessions held by VTAM in a recovery pending state are not always reestablished by CICS. CICS (or VTAM) unbinds recovery pending sessions in the following situations:

- If CICS does not restart within the specified persistent session delay interval
- If you perform a COLD start after a CICS failure
- If CICS restarts with XRF=YES (when the failed CICS was running with XRF=NO)

- If CICS cannot find a terminal control table terminal entry (TCTTE) for a session (for example, because the terminal was autoinstalled with AIRDELAY=0 specified)
- If a terminal or session is defined with the recovery option (RECOVOPT) set to UNCONDREL or NONE
- A connection is defined with the persistent session recovery option (PSRECOVERY) set to NONE.

In all these situations, the sessions are unbound, and the result is as if CICS has restarted following a failure without VTAM persistent session support.

There are some other situations where APPC sessions are unbound. For example, if a bind was in progress at the time of the failure, sessions are unbound.

Sessions not retained

There are some circumstances in which VTAM does not retain LU–LU sessions:

- VTAM does not retain sessions after a VTAM, MVS, or processor (CPC) failure
- VTAM does not retain CICS sessions if you close VTAM with any of the following CICS commands:
  - SET VTAM FORCECLOSE
  - SET VTAM IMMCLOSE
  - SET VTAM CLOSED
- VTAM does not retain CICS sessions if you close the CICS node with the VTAM command VARY NET INACT ID=applid
- VTAM does not retain CICS sessions if your CICS region performs a normal shutdown (with a PERFORM SHUTDOWN command).

Predatory takeovers

If persistent session support is in use, a VTAM application with the same APPLID as that of an executing CICS system can assume control of the sessions of the executing CICS system. This is known as a predatory takeover.

An example of a predatory takeover is where two CICS regions are running under the same APPLID, but only one has VTAM open with active sessions. If you issue a SET VTAM OPEN (CEMT or EXEC CICS) command, VTAM passes the sessions to the second CICS.

CICS responds to predatory takeovers of this sort by always unbinding existing sessions when the VTAM ACB is dynamically opened.

Comparison of persistent session support with XRF

XRF was introduced in CICS/MVS Version 2 to allow an alternate, partially initialized, CICS region to take over control from an active CICS system which had failed. However, XRF does not support LU6.2 connections and terminals, or locally-attached terminals. Furthermore, pairs of systems have to be defined and operated (instead of single systems), making system management more difficult.

Persistent session support overcomes these disadvantages of XRF. However, persistent session support does not retain sessions after VTAM, MVS, or CEC

failure. If you want protection from such failures, you should use XRF and not VTAM persistent sessions. Furthermore, a user does not remain signed on following a persistent session recovery.

Persistent session support has the following advantages over XRF:

- It supports all LU types except LU6.1 and LU0 pipeline sessions. XRF does not support local terminals or LU6.2 sessions.
- It is easier to install and manage than XRF, and requires only a single CICS region.

You can use the following combinations of system initialization parameters in CICS/ESA 4.1:

- **XRF=NO** and **PSDINT=0**. This provides availability through an emergency restart of a failed CICS region, without any XRF or VTAM persistent sessions support.
- **XRF=YES**. This provides availability of the system by an XRF takeover should the active CICS fail, and availability for the user by means XRF backup sessions.
- **XRF=NO** and **PSDINT>0**. This provides availability of the system through emergency restart-in-place, and availability for the end user by means of persistent session support. Only one set of datasets is required. Only one system is required. The system programmer needs to determine the recovery requirements of all terminals defined to CICS, and to determine a suitable persistent delay interval.

Benefits of persistent session support

The benefits of persistent session support fall into two categories:

- Benefits for terminal end users
- Benefits for CICSplexes.

Benefits for terminal end users

Without persistent session support, all sessions existing on a CICS system are lost when that CICS system fails. In any subsequent restart of CICS, the rebinding of sessions that existed before the failure depends on the terminal's AUTOCONNECT option. If AUTOCONNECT is specified for a terminal, the user of that terminal waits until the GMTRAN transaction has run before being able to continue working. If AUTOCONNECT is not specified for a terminal, the user of that terminal has no way of knowing (unless told by support staff) when CICS is operational again unless the user tries to log on. In either case, users are disconnected from CICS and need to reestablish a session, or sessions, to regain their working environment.

With persistent session support, sessions are put into recovery pending state on a CICS failure. If CICS starts within the specified interval, and RECOVOPTION is set to CLEARCONV or SYSDEFAULT, terminal users do not need to reestablish their session, or sessions, to regain their working environment. The sessions persist in a bound state, subject to the operation of the AIRDELAY system initialization parameter. This means that if a terminal is not used during the period specified on the AIRDELAY system initialization parameter its session is unbound.

The terminal user is notified of the successful recovery if MESSAGE is specified on RECOVNOTIFY of the TYPETERM resource definition.

Benefits for CICSplexes

Persistent session support improves availability of CICSplexes, particularly those with one or more terminal-owning regions that attach a large number of sessions.

It provides a faster restart of failed TORs and so improves the availability of an entire CICSplex. Without persistent session support, such a TOR may take a considerable time to restart after failure. If a CICSplex has only one TOR and it has failed, the entire CICSplex may become unavailable to end users. If a CICSplex has more than one TOR and one (or more) fails, large parts of the CICSplex may become unavailable to end users.

Requirements for persistent session support

For CICS single-node persistent sessions support, you need the VTAM persistent LU–LU session enhancements in VTAM 3.4.1 or later. CICS/ESA 4.1 functions with releases of VTAM earlier than 3.4.1, but in the earlier releases sessions are
+ not retained in a bound state in the event of a CICS failure. For multi-node
+ persistent sessions support you need VTAM 4.4.

See Chapter 39, “Prerequisite hardware and software for CICS/ESA 4.1” on page 651 for more information about supported releases of VTAM and NCP in CICS/ESA 4.1.

Resource usage

There is no significant change in real storage requirements nor in storage required below the 16MB boundary with persistent session support. However, during the recovery process, persistent session support requires 75KB of virtual storage, plus 125KB of virtual storage per 1000 sessions, above the 16MB boundary.

If there is insufficient contiguous storage for this, the process is not aborted but smaller areas of storage are used. This means that more iterations of the code are required and processing is slower. If there is insufficient storage, persistent session support is disabled, message DFHZC0003 is issued and an exception trace entry is made.

Changes to CICS externals

The introduction of persistent session support in CICS/ESA 4.1 results in a number of changes to CICS externals. These are:

- Changes to system definition
- Changes to resource definition
- Changes to the application programming interface
- Changes to the system programming interface
- Changes to global user exits
- Changes to user replaceable modules
- Changes to CICS-supplied transactions
- Changes to monitoring and statistics.

Changes to system definition

There is a new system initialization parameter, PSDINT, that specifies the persistent session delay interval. It indicates for how long sessions are to be held in recovery pending state following the failure of CICS.

Table 4 shows the syntax of the PSDINT system initialization parameter.

Table 4. The DFHSIT macro parameter for VTAM persistent sessions

	DFHSIT	TYPE={**CSECT**|DSECT} ... [,PSDINT={**0**|hhmmss}] [,PSTYPE={**SNPS**|MNPS}] ...
	END	DFHSITBA

PSDINT={0|hhmmss}

Specifies the persistent session delay interval. This delay interval specifies if, and for how long, VTAM is to hold sessions in a recovery-pending state if CICS fails. The value for hours can be in the range 0 through 23; the minutes and seconds in the range 00 through 59 inclusive.

This value can be overridden during CICS execution (and hence change the action taken by VTAM if CICS fails).

0 A zero value specifies that, if CICS fails, sessions are terminated. This is the default.

hhmmss

Specifies a persistent session delay interval from 1 second up to the maximum of 23 hours 59 minutes and 59 seconds. If CICS fails, VTAM holds sessions in recovery pending state for up to the interval specified on the PSDINT system initialization parameter.

Specify a 1-to-6 digit time in hours, minutes and seconds, up to the maximum time. If you specify less than six digits, CICS pads the value with leading zeros. Thus a value of 500 is taken as five minutes exactly.

The interval you specify must cover the time from when CICS fails to when the VTAM ACB is opened by CICS during the subsequent emergency restart.

PSTYPE={SNPS|MNPS}

Specifies whether you want CICS to use VTAM single-node or multi-node persistent sessions. Specify MNPS if VTAM multi-node persistent session support is installed (VTAM 4.4 or later), and you want to recover sessions when the VTAM ACB is re-opened, following a VTAM restart after an abend.

Note that MNPS does not recover APPC synclevel 2 sessions, which are unbound after CICS re-opens the VTAM ACB.

VTAM holds all sessions in recovery pending state for up to the interval specified (unless they are unbound through path failure or VTAM operator action, or other-system action in the case of intelligent LUs). The PSDINT value used must take account of the types and numbers of sessions involved.

You must exercise care when specifying large PSDINT values because of the problems they may give in some environments, in particular:

- Dial-up sessions—real costs may be incurred
- LU6.2 sessions to other host systems—such systems may become stressed.

Notes:

1. When specifying a PSDINT value, you must consider the number and, more particularly, the nature of the sessions involved. If LU6.2 sessions to other host systems are retained in recovery pending state, the other host systems may experience excessive queuing delays. This point applies to LU6.1 sessions which are retained until restart (when they are unbound).
2. The PSDINT parameter is incompatible with the XRF=YES parameter. If XRF=YES is specified, the PSDINT parameter is ignored.

Changes to resource definition

There are some changes to CICS resource definitions to support CICS persistent session support facility.

TYPETERM resource definition changes

Before CICS/ESA 4.1, the RECOVNOTIFY attribute applied to class 1 terminals only, and the RECOVOPTION attribute applied to class 1 and class 2 terminals, with both options having effect only on an XRF takeover. In CICS/ESA 4.1, these attributes are also used to indicate the actions to be taken by CICS/ESA following a CICS restart with VTAM persistent sessions. The distinction between class 1 and class 2 terminals is not made for persistent sessions restarts; however the following limitations exist for persistent sessions recovery:

- RECOVOPTION has no effect for LU0 pipeline and LU6.1 sessions. These sessions are always unbound following a persistent sessions restart.
- RECOVNOTIFY has no effect for LU6.2 sessions, or for sessions for which the recovery action (as determined by CICS, or as specified in the RECOVOPTION parameter) is to unbind the session.

RECOVNOTIFY({NONE|MESSAGE|TRANSACTION})

This option applies to the recovery of sessions for terminals in a CICS region running with either VTAM persistent sessions or with XRF. It is for use in situations where a terminal user may have to take action, such as sign on again, after a CICS restart. Use RECOVNOTIFY to specify how such a user should be notified.

VTAM persistent sessions: In a CICS region running with persistent session support, this specifies how a terminal end user is notified that their terminal session has been recovered.

XRF: In a CICS region running with XRF support, this specifies how the terminal user is notified that an XRF takeover has occurred.

This option is not applicable to APPC sessions.

NONE

There is no notification that recovery or a takeover has occurred.

MESSAGE

A message is displayed on the screen to say that the system has recovered. The message is specified in two BMS maps; DFHXRC1 and DFHXRC2 for XRF; and DFHXRC3 and DFHXRC4 for VTAM persistent sessions. These maps are in map set DFHXMSG. If reduced takeover time is important, use MESSAGE rather than TRANSACTION.

The terminal must be defined with the ATI(YES) option, and must be capable of displaying a BMS map.

TRANSACTION

A transaction is initiated at the terminal. The name of the transaction is specified by the RMTRAN system initialization parameter. (The default transaction for RMTRAN is the one specified in the GMTRAN system initialization parameter: the good-morning transaction.)

For the TRANSACTION operand, the terminal must be defined with the ATI(YES) option. For XRF only, if reduced takeover time is important, use MESSAGE rather than TRANSACTION.

RECOVOPTION({SYSDEFAULT|CLEARCONV|RELEASESESS|UNCONDREL|NONE})

This option applies to the recovery of sessions in a CICS region running with VTAM persistent sessions, or with XRF.

VTAM persistent sessions: In a CICS region running with persistent session support, this option specifies how you want CICS to recover the session, and return the terminal to service on system restart within the persistent session delay interval.

XRF: In a CICS region running with XRF support, this option specifies how you want CICS to recover the session, and return the terminal to service after an XRF takeover.

For all recovery options other than NONE, if the action taken is a VTAM UNBIND, the UNBIND is followed by a VTAM SIMLOGON.

SYSDEFAULT

VTAM persistent sessions: In a CICS region running with persistent session support, this specifies that CICS is to select the optimum procedure to recover a session on system restart within the persistent session delay interval, depending on the session activity and on the characteristics of the terminal.

Although sessions are recovered, any transactions in-flight at the time of the failure are abended and not recovered. Transactions are also abended if the recovered session is being used by another CICS region over an APPC connection.

CICS recovers the session with the least possible impact, in one of the following ways:

- If the terminal was not executing a transaction at the time of the CICS failure, no recovery action is required, and CICS takes the appropriate recovery notification action as defined by RECOVNOTIFY.
- If the terminal was busy (that is, executing a transaction) when CICS failed, CICS first tries to recover the session by sending a VTAM end-bracket indicator. If the end-bracket does not recover the session (for example, CICS may be in RECEIVE mode), CICS issues a CLEAR

command. If the terminal does not support the CLEAR command, the recovery action taken is a VTAM UNBIND followed by a SIMLOGON.

See the *CICS/ESA Recovery and Restart Guide* for more information about persistent sessions.

XRF: In a CICS region running with XRF support, this specifies that CICS is to select the optimum procedure to recover a busy session at takeover, depending on the session activity and on the characteristics of the terminal.

CLEARCONV
Prevents CICS from sending an end-bracket indicator to close an in-bracket session. Instead CICS sends a CLEAR request, to reset the conversation states. If the session does not support the CLEAR request, CICS sends an UNBIND request. The CLEAR or UNBIND is sent only if the session was busy at the time of system restart (in the case of persistent sessions) or the takeover (in the case of XRF).

RELEASESESS
Requires CICS to send an UNBIND request to release the active session. The UNBIND is sent only if the session was busy at the time of system restart (in the case of persistent sessions), or the takeover (in the case of XRF). Following the UNBIND, the session is queued for SIMLOGON. If the session is not busy, the requested recovery notification is carried out.

UNCONDREL
Requires CICS to send an UNBIND request to release the active session. The UNBIND is sent whether or not the session was busy at the time of system restart (in the case of persistent session support) or the takeover (in the case of XRF). Following the UNBIND, the session is queued for SIMLOGON.

NONE
VTAM persistent sessions: In a CICS region running with persistent session support, this specifies that the terminal session is not to be recovered at system restart within the persistent session delay interval: in effect, the terminal has no persistent session support. LU6.2 sessions are unbound but the latest negotiated CNOS value is returned to the CICS system after the restart. After system restart, the terminal is reconnected automatically if you specify AUTOCONNECT(YES), subject to the operation of the AIRDELAY system initialization parameter (AIRDELAY=0 overrides AUTOCONNECT(YES), and the terminal is not reconnected).

XRF: In a CICS region running with XRF support, this specifies that the logon state is not tracked by the alternate system, and the terminal session is not automatically recovered after a takeover; in effect, the terminal has no XRF support. After takeover, the terminal is reconnected automatically by the alternate system, if you specify AUTOCONNECT(YES).

CONNECTION resource definition changes

For LU6.2 sessions, the recovery option attribute is extended to the CONNECTION resource definition as the new attribute PSRECOVERY.

PSRECOVERY({SYSDEFAULT|NONE})
In a CICS region running with persistent sessions support, this specifies whether, and how, LU6.2 sessions are recovered on system restart within the persistent session delay interval.

SYSDEFAULT

If a failed CICS system is restarted within the persistent session delay interval, the following actions occur:

- User modegroups are recovered to the SESSIONS RECOVOPTION value.
- The SNASVCMG modegroup is recovered.
- The connection is returned in ACQUIRED state and the last negotiated CNOS state is returned.

NONE

All sessions are unbound as out-of-service with no CNOS recovery.

SESSIONS resource definition changes

The RECOVNOTIFY option does not apply to sessions, and has been removed in CICS/ESA 4.1.

Before CICS/ESA 4.1, RECOVOPTION applied to XRF, and specified whether the sessions was tracked and reconnected at restart time. At CICS/ESA 4.1, RECOVOPTION also applies to VTAM persistent sessions support, and determines whether the session persists or not.

RECOVOPTION({SYSDEFAULT|CLEARCONV|
RELEASESESS|UNCONDREL|NONE})

This option applies to the recovery of sessions in a CICS region running with VTAM persistent sessions, or with XRF.

VTAM persistent sessions: In a CICS region running with persistent session support, this option specifies how you want CICS to recover the session, and return the terminal to service on system restart within the persistent session delay interval.

XRF: In a CICS region running with XRF support, this option specifies how you want CICS to recover the session, and return the terminal to service after an XRF takeover.

For all recovery options other than NONE, if the action taken is a VTAM UNBIND, the UNBIND is followed by a VTAM SIMLOGON.

SYSDEFAULT

VTAM persistent sessions: In a CICS region running with persistent session support, this specifies that CICS is to select the optimum procedure to recover a session on system restart within the persistent session delay interval, depending on the session activity and on the characteristics of the terminal.

Although sessions are recovered, any transactions in-flight at the time of the failure are abended and not recovered. Transactions are also abended if the recovered session is being used by another CICS region over an APPC connection.

CICS recovers the session with the least possible impact, in one of the following ways:

- If the session was not busy at the time that CICS/ESA failed, no action is required.

- If the session was busy at the time that CICS/ESA failed, CICS issues a DEALLOCATE(ABEND) (equivalent to an EXEC CICS ISSUE ABEND) for the APPC conversation in progress at the time of the failure.
- If neither of the above applies, the session is unbound.

XRF: If AUTOCONNECT(YES) is specified, the session is restarted. If AUTOCONNECT(NO) is specified, the session is unbound.

CLEARCONV

VTAM persistent sessions: CLEARCONV is not supported for APPC sessions. It defaults to SYSDEFAULT.

XRF: If AUTOCONNECT(YES) is specified, the session is restarted. If AUTOCONNECT(NO) is specified, the session is unbound.

RELEASESESS

VTAM persistent sessions: CLEARCONV is not supported for APPC sessions. It defaults to SYSDEFAULT.

XRF: If AUTOCONNECT(YES) is specified, the session is restarted. If AUTOCONNECT(NO) is specified, the session is unbound.

UNCONDREL

Requires CICS to send an UNBIND request to release the active session. The UNBIND is sent whether or not the session was busy at the time of system restart (in the case of persistent session support) or the takeover (in the case of XRF).

NONE

VTAM persistent sessions: In a CICS region running with persistent session support, this specifies that the session is not to be recovered at system restart within the persistent session delay interval: in effect, the sessions on the modegroup have no persistent session support. LU6.2 sessions are unbound and the modegroup CNOS value is reset to zero. After system restart, the session is reconnected automatically if you specify AUTOCONNECT(YES).

XRF: In a CICS region running with XRF support, this specifies that the logon state is not tracked by the alternate system, and the terminal session is not automatically recovered after a takeover; in effect, the terminal has no XRF support. After takeover, the terminal is reconnected automatically by the alternate system, if you specify AUTOCONNECT(YES).

Changes to the system programming interface

General-use programming interface

The CICS system programming interface (SPI) is extended with changes to the following commands:

- EXEC CICS INQUIRE VTAM
- EXEC CICS SET VTAM

The INQUIRE VTAM command tells you the state of the connection between your CICS system and VTAM. The OPENSTATUS option is now optional.

Four new keywords are added to represent the persistent session delay interval.

The SET VTAM command opens and closes the VTAM ACB and, in CICS/ESA 4.1, allows you to specify if and for how long sessions are to be kept in recovery pending state after a CICS failure. The OPENSTATUS option is now optional.

Four new keywords are added to allow the setting of the PSDI.

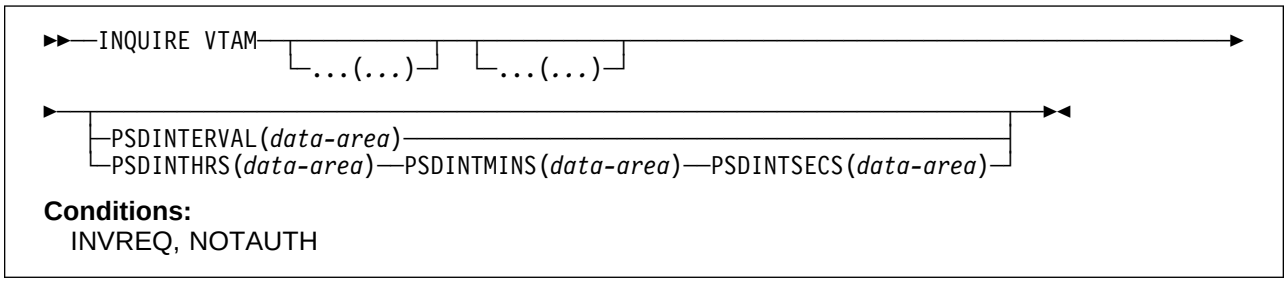
Note: Data-area and data-value fields are fullword binary values.

EXEC CICS INQUIRE VTAM command

Function

Inquire on the state of the connection between CICS and VTAM and also on the persistent session delay interval (PSDI).

Syntax



Conditions:
INVREQ, NOTAUTH

INQUIRE VTAM options

PSDINTERVAL(data-area)

returns, as a 4-byte packed-decimal value, the persistent session delay interval (PSDI) in the form "**0hhmmss+**". It specifies if and for how long sessions are held in recovery-pending state after a CICS failure. The time interval can be in the range 00 to 23 hours; 00 to 59 minutes; and 00 to 59 seconds.

Note: If you do not use this option but prefer to use the individual PSDINTHRS, PSDINTMINS, and PSDINTSECS options instead, you must use all three individual options (PSDINTHRS, PSDINTMINS, and PSDINTSECS).

PSDINTHRS(data-area)

returns, as a full-word binary value, the number of hours for which sessions are to be kept in recovery pending state after a CICS failure (the PSDI value). Values are in the range 0 to 23.

PSDINTMINS(data-area)

returns, as a full-word binary value, the number of minutes beyond the hour value specified in PSDINTHRS for which sessions are to be kept in recovery pending state following a CICS failure (the PSDI value). Values are in the range 0 to 59.

PSDINTSECS(data-area)

returns, as a full-word binary value, the number of seconds beyond the minute value specified in PSDINTMINS for which sessions are to be kept in recovery pending state following a CICS failure. Values are in the range 0 to 59.

The OPENSTATUS option remains unchanged from CICS/ESA 3.3.

INQUIRE VTAM conditions

INVREQ
occurs either if PSDINTERVAL or PSDINTHRS, PSDINTMINS, PSDINTSECS has been specified and VTAM does not support persistent sessions, or if VTAM is not present in the system.

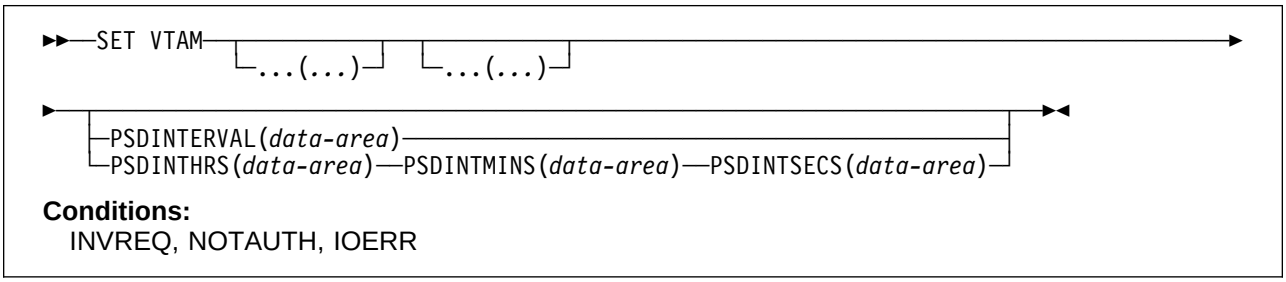
NOTAUTH
occurs if use of this command is not authorized.

EXEC CICS SET VTAM command

Function

Open/close the VTAM ACB and set the persistent session delay interval.

Syntax



SET VTAM options

PSDINTERVAL(data-value)
specifies, as a 4-byte packed-decimal value, the persistent session delay interval (PSDI) in the form "**0hhmmss+**". It specifies if and for how long sessions are held in recovery-pending state after a CICS failure. The time interval can be in the range 00 to 23 hours; 00 to 59 minutes; and 00 to 59 seconds.

If 0 is specified, sessions are terminated at the time of failure and do not persist.

A value of –1 causes the option to be ignored and the current value is unchanged.

If you specify this option and also the individual hours, minutes, and seconds options, CICS accepts the individual time components and ignores PSDINTERVAL.

PSDINTHRS(data-value)
specifies, as a full-word binary value, the number of hours for which sessions are to be kept in recovery pending state after a CICS failure (the PSDI value). The permitted values are in the range 0 to 23. The default is 0.

PSDINTMINS(data-value)
specifies, as a full-word binary value, the number of minutes beyond the hour value specified in PSDINTHRS for which sessions are to be kept in recovery pending state following a CICS failure (the PSDI value). The permitted values are in the range 0 to 59. The default is 0.

PSDINTSECS(data-value)
specifies, as a full-word binary value, the number of seconds beyond the minute value specified in PSDINTMINS for which sessions are to be kept in recovery pending state following a CICS failure. The permitted values are in the range 0 to 59. The default is 0.

+ **OPEN**
+ If CICS is using VTAM multi-node persistent sessions, and VTAM has been
+ restarted after an abend, opening the VTAM ACB causes CICS to restore the
+ persistent sessions that VTAM has retained. However, CICS does not restore
+ APPC synclevel 2 sessions, which are unbound.

If all three options, PSDINTHRS, PSDINTMINS, and PSDINTSECS are specified as having the value –1, the function is ignored and the current interval is unchanged.

CICS passes the specified value to VTAM if both the following conditions are met:

- The VTAM ACB is open, or has just been opened by OPENSTATUS(OPEN) being specified on the SET VTAM command.
- CICS can exploit persistent sessions (it is running with VTAM 3.4.1 or later).

If the VTAM ACB is closed or is being closed (by OPENSTATUS CLOSED, or IMMCLOSE, or FORCECLOSE) the specified interval value is saved and passed to VTAM when the ACB is next opened. Should the OPEN command fail, the interval value is saved for passing to VTAM on the next successful OPEN command. The OPENSTATUS option remains unchanged from CICS/ESA 3.3.

SET VTAM conditions

Condition	RESP2	Meaning
INVREQ	1	VTAM is not present in the system.
INVREQ	2	OPENSTATUS has an invalid CVDA value.
INVREQ	4	PSDINTERVAL value is not in the range 000000-235959.
INVREQ	5	PSDINTHRS value is not in the range 0-23.
INVREQ	6	PSDINTMINS value is not in the range 0-59.
INVREQ	7	PSDINTSECS value is not in the range 0-59.
INVREQ	8	Attempted to set PSDINTERVAL > 0 in an XRF-capable system.
INVREQ	9	An error occurred during attempt to set persistent session delay interval. Check the console for any DFHZCnnnn messages that relate to this error.
INVREQ	10	Attempted to set PSDINTERVAL > 0, but using a TCT generated against, or running with, a level of VTAM not capable of persistence. The ACB has been OPENED successfully, and the PSDINTERVAL will have been reset to 0. Either of the following happened: • The command specified PSDINTERVAL > 0. • The command specified PSDINTHRS or PSDINTMINS or PSDINTSECS as > 0. • An earlier command specified one of the above as > 0 whilst the ACB was closed/being closed.

Condition	RESP2	Meaning
INVREQ	11	The ACB has opened successfully, but in attempting to unbind sessions that persisted, an error occurred. CICS saves the interval value to pass to VTAM on the next successful OPEN ACB. Check the console for any DFHZCnnnn messages that relate to this error.
INVREQ	12	The ACB has been closed whilst this command was being processed. CICS saves the interval value to pass to VTAM on the next successful OPEN ACB. Check the console for any DFHZCnnnn messages that relate to this error. Retry this command if appropriate.
INVREQ	13	An error occurred during recovery of sessions, and the VTAM ACB will subsequently have been closed. CICS saves the interval value to pass to VTAM on the next successful OPEN ACB. Check the console for any DFHZCnnnn messages that relate to this error.
IOERR	3	An error occurred during the opening of the ACB. CICS saves the interval value to pass to VTAM on the next successful OPEN ACB.
IOERR	¬ 3	An error occurred during the opening of the ACB. RESP2 value is in register 15 or FDBK2 value from VTAM. CICS saves the interval value to pass to VTAM on the next successful OPEN ACB.
NOTAUTH	100	The use of this command is not authorized.

Note: SET VTAM OPEN is not supported by CICS as a way of taking over a network.

Changes to the application programming interface

- Persistent session support does not preserve VTAM logon data.
- There is a new reason for a SYSBUSY condition an EXEC CICS ALLOCATE (APPC) command.

Loss of VTAM logon data

VTAM logon data is not retained over a restart. Any logon data which has not been extracted and saved (for example, in a recoverable temporary storage queue) is lost. You can use the EXEC CICS EXTRACT LOGONMSG command to obtain and save this data.

SYSBUSY condition on EXEC CICS ALLOCATE (APPC)

A SYSBUSY condition can be returned on an EXEC CICS ALLOCATE command. This can occur if the ALLOCATE command is issued when persistent session recovery is still in process, and the sessions needed to satisfy the command are not yet recovered.

End of General-use programming interface

Changes to the XXRSTAT global user exit

Product-sensitive programming interface

In CICS/ESA 3.3, XXRSTAT global user exit programs are invoked after a VTAM failure, allowing you to decide whether CICS should terminate or continue after the failure. In CICS/ESA 4.1, persistent session support XXRSTAT can also be invoked in the event of a predatory takeover.

If persistent session support is in use, a VTAM application with the same APPLID as that of an executing CICS region can assume control of all the sessions of the executing CICS region. This is known as a predatory takeover, and XXRSTAT is invoked to enable you to choose between allowing CICS (the system which has suffered the takeover) to continue or terminate.

If your XXRSTAT user-exit program exits with UERCNORM, to indicate that CICS is to take the default system action, CICS abends without a dump if a predatory takeover is the cause of invoking your exit. There is also a return code (UERCOIG) that means CICS is to continue running without VTAM support, and the predatory application assumes control of all of the VTAM sessions. However, this is not recommended. In the event of a predatory takeover, any TCAM sessions remain bound to CICS, and with the VTAM sessions acquired by the predator this could cause integrity problems.

To safeguard your CICS regions against predatory takeovers, you are recommended to RACF-protect your CICS APPLIDs, as described in the *CICS/ESA CICS-RACF Security Guide*.

If you require CICS to terminate, it terminates with the abend code specified in your XXRSTAT exit program.

Changes to user-replaceable modules

With persistent session support, there are considerations for the node error program (NEP).

One of the steps in the conversation-restart process is to link to the node error program with error code X'FD'. If you want to be able to change any of the system-wide recovery notification options (whether terminal users are notified of a recovery, the recovery message, or the recovery transaction) for some terminals, you should write your own error processor to handle code X'FD'. (For details of the recovery notification parameters passed to the NEP, see the *CICS/ESA Customization Guide*.)

When using persistent sessions, note the following:

- When a session has been recovered, it may be a good idea to run NEP processing equivalent to your normal “session started” (code X'48') processing, because code X'48' is not passed on session recovery when persistent sessions are used.
- In certain situations where sessions have persisted over a failure but have been unbound on restart (for example, a COLD start occurs after a CICS failure), the NEP is not driven. (In systems without persistent session support, the NEP is always driven with code X'49', “session terminated”, when a VTAM session terminates.) Conditions leading to the issuing of the following

messages do not drive the NEP. The messages appear on the system console:

```
DFHZC0120     DFHZC0122     DFHZC0124     DFHZC0130
DFHZC0121     DFHZC0123     DFHZC0129
```

Conditions leading to the issuing of messages DFHZC0125 and DFHZC0131 drive the NEP with codes X'FB' and X'FC' respectively. It is recommended that you run NEP processing equivalent to your normal “session terminated” NEP processing for these conditions.

- If zero is specified on the AIRDELAY system initialization parameter, autoinstalled terminals are not recovered after a restart. Similarly, if the delay period specified on AIRDELAY expires before an autoinstalled terminal is used after a restart, the terminal definition is deleted. In these circumstances, any expected NEP processing as a result of CLSDSTP=NOTIFY being coded does not take place.

End of Product-sensitive programming interface

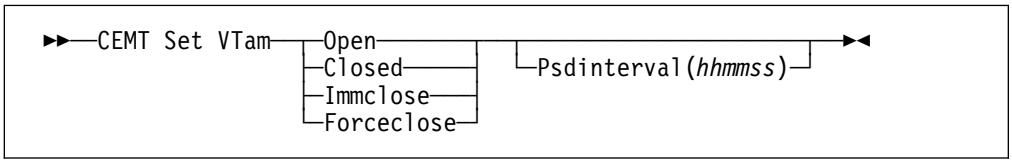
Changes to CICS-supplied transactions

CEMT INQUIRE VTAM and CEMT SET VTAM commands are changed to include the PSDINTERVAL option.

CEMT INQUIRE|SET VTAM

The INQUIRE VTAM command returns the value of the persistent session delay interval. The SET VTAM command allows you to change the PSDINTERVAL
+ value. Also the function of the OPEN option is extended when CICS is using
+ VTAM with mutli-node persistent sessions.

Command syntax



Command options

Psdinterval

displays the persistent session delay interval (PSDI) in the form “**0hhmmss+**”. It specifies if and for how long sessions are held in recovery-pending state after a CICS failure. You can reset this value by overtyping it with a different value. The time interval can be in the range 00 to 23 hours; 00 to 59 minutes; and 00 to 59 seconds.

+ **Open**

+ If CICS is using VTAM multi-node persistent sessions, and VTAM has been
+ restarted after an abend, opening the VTAM ACB causes CICS to restore the
+ persistent sessions that VTAM has retained. However, CICS does not restore
+ APPC synclevel 2 sessions, which are unbound.

CEMT issues the following error messages in response to invalid attempts to set the persistent session delay interval:

```
INVALID PSDINT
NOT WITH XRF
SETLOGON FAILURE
BACK LEVEL VTAM
ACB CLOSED
RECOVERY ERROR
```

Changes to monitoring and statistics

The following statistics are collected:

- PS_NIB_COUNT = the number of VTAM sessions that persisted.
- PS_INQUIRE_COUNT = the number of times CICS issued INQUIRE OPTCD=PERSESS.
- PS_UNBIND_COUNT = the number of persisting sessions that were terminated.
- PS_OPNDST_COUNT = the number of persisting sessions that were successfully restored.
- PS_ERROR_COUNT = the number of persisting sessions that were already unbound when CICS tried to restore them.

These are treated in the same way as the other VTAM statistics (such as A03ADOC) and are used as message inserts in some of the persistent session support generated messages.

Using FEPI with VTAM persistent sessions

When creating FEPI applications, you need to be aware of the possible effects of the use of VTAM persistent sessions in the front- or back-end systems. For information about support for VTAM persistent session in CICS/ESA 4.1, see the *CICS/ESA Recovery and Restart Guide*.

Restart of front-end system using persistent sessions

Using persistent sessions in the front-end does not give FEPI any additional recoverability benefits. FEPI is always cold started; thus, to FEPI, the effect of restarting a front-end system for which persistent sessions support is enabled is indistinguishable from a cold start of CICS.

Restart of back-end system using persistent sessions

In the back-end system, there are terminal definitions that are used when the FEPI simulated terminals establish sessions with the target. These definitions may be hard-coded, or may be autoinstall model definitions. If the terminal definitions have been set up to use persistent session support, and the back-end system is restarted within the persistent session delay interval, the terminal sessions are recovered.

Effect on FEPI application programs

It is likely that FEPI application programmers have little say in the way that persistent session support is used in the back-end system. They therefore need to be aware of the different ways in which terminal sessions can be recovered, so that their applications cater for all possibilities. If the back-end (target) is a CICS/ESA 4.1 system, the way in which a session is recovered depends on the setting of the RECOVOPTION and RECOVNOTIFY options of the TYPETERM definition.

RECOVOPTION(SYSDEFAULT)

On restart within the persistent session delay interval, CICS selects the optimum procedure to recover a session.

For LU2, if the session is busy and CICS is in send mode, CICS sends an end bracket. If the session is busy and CICS is not in send mode, CICS sends an SNA CLEAR request to reset the conversation state.

If a FEPI conversation is in progress when the target system terminates, your application could see one of the following:

- A timeout on a RECEIVE, CONVERSE, or START command, while it waits for the target to restart.

  Deal with this in the normal way for a timeout.

- A FEPI RECEIVE or CONVERSE command completes as a result of the end bracket sent by CICS. The RU on this data flow may be empty or may contain a user-defined message, depending on the value of the RECOVNOTIFY option.

  Your application may need to perform some backout processing.

- An INVREQ response with a RESP2 value of 230 on a FEPI SEND, RECEIVE, CONVERSE, ISSUE, or START command, indicating that an SNA CLEAR was received.

  Your application may need to perform some backout processing.

You must also consider the value specified for RECOVNOTIFY:

RECOVNOTIFY(MESSAGE)

A message (defined in the BMS maps DFHXRC3 and DFHXRC4) is sent to the “terminal”. Your FEPI application must contain logic to deal with this data flow.

If there is no active conversation at the time of restart, the flow is received as unsolicited data at the FEPI front-end.

RECOVNOTIFY(TRANSACTION)

A transaction is initiated in the target. The default is the Good Morning transaction. Your application must contain logic to deal with this data flow.

If there is no active conversation at the time of restart, the flow is received as unsolicited data at the FEPI front-end.

RECOVNOTIFY(NONE)

The “terminal” is not notified that a restart has occurred. Your application need take no special action.

RECOVOPTION(CLEARCONV)

On restart within the persistent session delay interval, CICS sends an SNA CLEAR request to reset the conversation states. The CLEAR is sent only if the session was busy at the time of system restart. If a FEPI conversation is in progress when the target system terminates, your application could see one of the following:

- A timeout on a RECEIVE, CONVERSE, or START command, while it waits for the target to restart.

  Deal with this in the normal way for a timeout.

- An INVREQ response with a RESP2 value of 230 on a FEPI SEND, RECEIVE, CONVERSE, ISSUE, or START command, indicating that an SNA CLEAR was received.

  Your application may need to perform some backout processing.

You must also consider the value specified for RECOVNOTIFY. The possible values are as described above, for RECOVOPTION(SYSDEFAULT).

RECOVOPTION(RELEASESESS)

On restart within the persistent session delay interval, CICS sends an UNBIND request to release an active session. The request is sent only if the session was busy at the time of system restart.

If a FEPI conversation is in progress when the target system terminates, your application could see one of the following:

- A timeout on a RECEIVE, CONVERSE, or START command, while it waits for the target CICS to restart.

  Deal with this in the normal way for a timeout.

- An INVREQ response with a RESP2 value of 215 on any FEPI command, indicating a 'session lost' condition.

  Deal with this in the normal way for a session loss.

RECOVOPTION(UNCONDREL)

On restart within the persistent session delay interval, CICS sends an UNBIND request to release an active session. The request is sent whether or not the session was busy at the time of system restart.

If a FEPI conversation is in progress when the target system terminates, your application could see either of the symptoms described for RECOVOPTION(RELEASESESS).

RECOVOPTION(NONE)

Even if the system is restarted within the persistent session delay interval, the session is not recovered—it has no persistent session support.

Deal with this in the normal way for a session loss.

Chapter 4. Intersystem session queue management

This chapter describes the new connection definition parameters and the new global user exit, XZIQUE, introduced in CICS/ESA 4.1 to enable you to manage the number of queued requests for sessions (allocate queues). It covers the following topics:

- Overview
- Benefits of using the global user exit
- Changes to CICS externals
- Designing an XZIQUE global user exit program.

Overview

In a perfect intercommunication environment, queues would never occur because work flow would be evenly distributed over time, and there would be enough intersystem sessions available to handle the maximum number of requests arriving at any one time. However, in the real world this is not the case, and, with peaks and troughs in the workload, queues do occur: queues come and go in response to the workload. The situation to avoid is an unacceptably high level of queuing that causes a bottleneck[1] in the work flow between interconnected CICS regions, and which leads to performance problems for the terminal end-user as throughput slows down or stops. This abnormal and unexpected queuing should be prevented, or dealt with when it occurs: a “normal” or optimized level of queuing can be tolerated.

For example, function shipping requests between CICS application-owning regions and connected file-owning regions can be queued in the issuing region while waiting for free sessions. Provided a file-owning region deals with requests in a responsive manner, and outstanding requests are removed from the queue at an acceptable rate, then all is well. But if a file-owning region is unresponsive, the queue can become so long and occupy so much storage that the performance of connected application-owning regions is severely impaired. Further, the impaired performance of the application-owning region can spread to other regions. This condition is sometimes referred to as “sympathy sickness”, although it should more properly be described simply as intersystem queuing, which, if not controlled, can lead to performance degradation across more than one region.

Internal CICS solution

CICS provides an internal solution based on values you specify for the QUEUELIMIT and MAXQTIME parameters of the connection resource definition. If you specify values for these parameters, CICS restricts any queues on the connection using the limits specified.

See “Changes to resource definition” on page 73 for information about the operation of these parameters.

In addition CICS provides the XZIQUE exit point to allow you to handle exceptional queue conditions in your own way.

[1] Bottleneck. A condition or state of the connection that slows down the rate of flow of traffic across the intercommunication link.

The XZIQUE global user exit

A new global user exit, XZIQUE, is introduced in CICS/ESA 4.1 to enable a user-written exit program to detect queuing problems (bottlenecks) early. CICS/ESA 3.3 introduced the XISCONA global user exit to allow the queuing of function shipping requests to be limited. The XZIQUE exit extends the function provided by XISCONA, which is driven only for function shipping requests. XZIQUE allows you to deal with allocate requests originating from function shipping, transaction routing, and other forms of intercommunication request. This provides increased flexibility to help you to handle bottlenecks caused by these requests. An XZIQUE global user exit program can respond to situations in ways most appropriate for the special circumstances of each customer installation.

The exit enables allocate requests to be queued or rejected, depending on the length of the queue. The exit also allows a connection on which there is a bottleneck to be cleared of its backlog.

Interaction with the XISCONA exit

The XISCONA exit continues to be supported for compatibility purposes in this release.

There is no interaction between the XZIQUE and XISCONA global user exits. If you enable both exits, XISCONA and XZIQUE could both be driven for function shipping requests, which is not recommended. You should ensure that only one of these exits is enabled.

You can use an XISCONA global user exit program at the XZIQUE exit point, suitably modified to use the XZIQUE parameter list.

Additions to connection definition

There are two new attributes added to the CONNECTION resource definition. For those situations where simple control requirements are adequate, you can specify the ‘queue limit’ parameter for connections. If you also need to detect and react quickly to bottlenecks, you can use the ‘maximum queue time’ parameter.

If these parameters are not sufficient for your installation, you can write an XZIQUE global user exit program to customize how you control bottlenecks on sessions.

Benefits of using the global user exit

A XZIQUE global user exit program can be written to detect bottlenecks early and take actions that might resolve a problem before the situation becomes really serious and begins to affect other interconnected regions. Detecting undesirable queuing early can help to improve reliability and availability in busy communicating systems.

Changes to CICS externals

The changes introduced in CICS/ESA 4.1 to allow flexible control of the queuing of allocate requests result in some enhancements to CICS externals: These are:

- Changes to resource definition
- Changes to the application programming interface
- Changes to the global user exit interface.

Changes to resource definition

There are two new parameters on the DEFINE CONNECTION resource definition command to allow you to specify controls for the queuing of allocate requests. These are the QUEUELIMIT and MAXQTIME parameters added to the CONNECTION PROPERTIES set of options, as in the partial CEDA DEFINE panel illustrated in Figure 5.

```
DEFINE CONN(HAA1) GR(CONNHAA#)
OVERTYPE TO MODIFY                                    CICS RELEASE = 0410
 CEDA  DEFine
  COnnection     : HAA1
  Group          : Connection to Hursley AOR1
  DEscription  ==>
 CONNECTION IDENTIFIERS
   ...
 REMOTE ATTRIBUTES
   ...
 CONNECTION PROPERTIES
  ACcessmethod ==> Vtam              Vtam | IRc | INdirect | Xm
  Protocol     ==> Appc              Appc | Lu61
  SInglesess   ==> No                No | Yes
  DAtastream   ==> User              User | 3270 | SCs | STrfield | Lms
  RECordformat ==> U                 U | Vb
  Queuelimit   ==> No                No | 0-9999
  Maxqtime     ==> No                No | 0-9999
 OPERATIONAL PROPERTIES
  ...
 SECURITY
  ...

                                                      APPLID=CICSHAA1
                                          TIME:  12.50.19  DATE: 93.007
PF 1 HELP 2 COM 3 END       6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 5. The DEFINE CONNECTION options

QUEUELIMIT

The maximum number of allocate requests that CICS is to queue while waiting for free sessions. You can specify either NO, or a number:

NO

There is no limit set to the number of allocate requests that CICS can queue while waiting for a free session. In this case, a value of X'FFFF' is passed on the XZIQUE parameter list (in field UEPQUELM).

number

The maximum number of allocate requests, in the range 0 through 9999, that CICS can queue on the connection while waiting for a free session. When the number of queued allocate requests reaches this limit, CICS rejects subsequent allocate requests until the queue drops below the limit.

This queue limit is passed to an XZIQUE global user exit program on the XZIQUE parameter list if the exit is enabled.

You can also control the queuing of allocate requests through the MAXQTIME parameter, and through an XZIQUE global user exit program. See the MAXQTIME parameter for more information about controlling queues.

MAXQTIME

A time control on the wait time of queued allocate requests that are waiting for free sessions on a connection that appears to be unresponsive. The maximum queue time is used only if a queue limit is specified on QUEUELIMIT, and then

the time limit is applied only when the queue length has reached the QUEUELIMIT value.

You can specify either NO, or a number:

NO
CICS maintains the queue of allocate requests that are waiting for a free session. No time limit is set for the length of time requests can remain queued (though the DTIMOUT mechanisms can apply to individual requests). In this case, a value of X'FFFF' is passed on the XZIQUE parameter list (in field UEPEMXQT).

number
The approximate upper limit on the time that allocate requests can be queued for a connection that appears to be unresponsive. The number represents seconds in the range 0 through 9999.

CICS uses the maximum queue time parameter to control a queue of allocate requests waiting as follows:

- When the number of queued allocate requests reaches the queue limit (QUEUELIMIT), and
- A new allocate request is received for the connection, and
- The rate of processing for the queue indicates that, on average, the new allocate will take more than the maximum queue time, then
- The queue is purged, and message DFHZC2300 is issued.

No further queuing takes place until the connection has successfully freed a session. At this point, CICS issues DFHZC2301 and resumes normal queuing.

You can also control the queuing of allocate requests through an XZIQUE global user exit program. This allows you more flexibility to use statistics provided by CICS, which report the state of the link. You can use these statistics, in combination with the queue limit and maximum queue time values you specify, to make more specialized decisions about queues.

The MAXQTIME value is passed to an XZIQUE global user exit program on the XZIQUE parameter list, if the exit is enabled. See the *CICS/ESA Customization Guide* for information about writing an XZIQUE global user exit program.

You can also specify the NOQUEUE|NOSUSPEND option on the ALLOCATE command to prevent an explicit request being queued. See the *CICS/ESA Application Programming Reference* for information about these API options.

Changes to the application programming interface

General-use programming interface

The EXEC CICS ALLOCATE command is unchanged, but it can now fail if there are no sessions available to satisfy the request and the QUEUELIMIT or MAXQTIME parameters, or a XZIQUE global user exit program, decides the request should not be queued. If CICS or the global user exit program rejects the request, CICS returns the SYSIDERR condition to the application program that issued the ALLOCATE request. Note that there are no separate RESP2 values to

distinguish between SYSIDERR conditions returned from the queue control mechanisms and other SYSIDERR conditions, but further information is provided in EIBRCODE. Your application can test EIBRCODE to determine whether the allocate failed because of action taken by the queue control mechanisms. The information in bytes 1 and 2 of EIBRCODE is as follows:

```
.. 08 00 .. .. ..   no session available; all sessions are
                   out of service, or released, or being quiesced.

.. 08 04 .. .. ..   no session available; request to queue
                   rejected by CICS due to the QUEUELIMIT parameter,
                   or by an XZIQUE global user exit program.
```

If CICS rejects a request with `08 04` in EIBRCODE, it also causes the COMMAREA of the dynamic transaction routing program to be updated in the terminal-owning region. CICS sets the DYRERROR parameter to a value of 3 to indicate to the dynamic routing program that the allocate request has been rejected.

SYSIDERR is also returned to waiting application programs if their requests are subsequently canceled after having first been queued. This can happen if the XZIQUE global user exit program returns to CICS with a return code that causes queued requests to be canceled. In this case, the information in bytes 1 and 2 of EIBRCODE is as follows:

```
.. 08 08 .. .. ..   no session available; a request which had
                   been queued was purged by the action of CICS using
                   the MAXQTIME parameter or by an XZIQUE global user
                   exit program.
```

If CICS rejects a request with `08 08` in EIBRCODE, it also causes the COMMAREA of the dynamic transaction routing program to be updated in the terminal-owning region. CICS sets the DYRERROR parameter to a value of 4 to indicate to the dynamic routing program that not only has the allocate request been rejected, but that the queue of outstanding requests has been purged.

See “Using the XZIQUE global user exit” on page 76 for details.

End of General-use programming interface

Changes to the global user exit interface

Product-sensitive programming interface

The exit programming interface is not changed: the new XZIQUE exit obeys the same rules as other global user exits.

XZIQUE is driven, if it is enabled, for two reasons:

1. Whenever CICS tries to allocate a session with a remote system and there is no free session available. CICS invokes the global user exit before queuing the allocate request, and the exit program can indicate through a return code what CICS should do next. This includes rejecting the allocate request, or suppressing future attempts to queue because the remote system is judged to be completely unresponsive.

2. After queuing has been suppressed, when an allocate request succeeds in finding a free session. In this case, the exit program is given the opportunity to resume queuing.

Requests for sessions can arise in a number of ways, such as explicit EXEC CICS ALLOCATE commands issued by application programs, or by transaction routing or function shipping requests.

XZIQUE is enabled by the EXEC CICS ENABLE command, and is disabled by the EXEC CICS DISABLE command.
The exit details are summarized as follows:

Table 5. Summary of global user exit

Exit name	Module or domain	Where or when invoked
XZIQUE	Allocate/free sessions program	1. Whenever an ALLOCATE request that cannot be satisfied is about to be queued, and 2. Whenever an ALLOCATE succeeds following the suppression of queuing as a result of a previous call to the exit program. **Note:** This exit is not invoked if the allocate request specifies NOQUEUE or NOSUSPEND.

Using the XZIQUE global user exit

When the exit is enabled, the global user exit program is able to check on the state of the allocate queue for a particular connection in the local system. Information about the state of the allocate queue against a connection is passed to an XZIQUE global user exit program in the exit-specific parameter list. Note that the parameter list is structured to provide data about non-specific allocate requests, or specific modegroup allocate requests, depending on the session request. Non-specific allocate requests are for MRO, LU6.1, and APPC sessions that do not specify a modegroup.

Using the information passed in the parameter list, your global user exit program can decide (based on queue length, for example) whether CICS is to queue the allocate request. Your program communicates its decision to CICS by means of one of the return codes CICS provides. These are:

UERCAQUE

This return code indicates that CICS is to queue the allocate request.

The total number of allocate requests queued against the connection is provided in field A14ESTAQ of the system entry statistics (for all non-specific allocates) or A20ESTAQ of the mode entry statistics (for specific modegroup allocates). See DSECTs DFHA14DS or DFHA20DS for details. CICS passes to the exit program, in the exit specific parameter UEPQUELIM, the QUEUELIMIT parameter from the connection definition.

If the limit has not been reached, you can return control to CICS with return code UERCAQUE.

UERCAPUR

This return code indicates that CICS is to reject the allocate request and return SYSIDERR to the application program, but leave the existing queue unchanged.

If the number of queued allocate requests has reached the limit set on the QUEUELIMIT parameter for the connection, you can request that CICS rejects the request. However, you should first check whether the state of the link is satisfactory. This means checking that the rate of allocation of sessions is acceptable. Use the time the queue was started, the current time, and the total number of allocates processed since the queue began, to determine the rate at which CICS is processing requests. The relevant fields are: UEPSAQTS and UEPSACNT for non-specific allocate requests; and UEPMAQTS and UEPMACNT for specific modegroup requests.

You can compare the calculated time with either:

1. The parameter from the connection definition, MAXQTIME, which is passed in the exit specific parameter UEPEMXQT, or
2. Some other preset time value

to determine whether CICS is allocating requests for sessions on this connection at an acceptable rate. If the processing time using this kind of formula is acceptable, return control to CICS with return code UERCAPUR to purge only this request.

UERCAKLL or UERCAKLM

These return codes indicate that you want CICS to deal with the request as follows:

- UERCAKLL—reject this request, purge all other queued allocate requests on this connection, and send an information message to the operator console.
- UERCAKLM—reject this request, purge all other queued modegroup allocate requests on this connection, and send an information message to the operator console.

If the queue limit has been reached but the performance of allocate processing against the queue is below the acceptable limits defined in your user exit program, you can return control to CICS as follows:

- For non-specific allocate requests, use return code UERCAKLL. UERCAKLL also returns SYSIDERR to all application programs waiting on the purged allocate requests. CICS sets the UEPFLAG parameter to UEPRC8 on subsequent calls to your XZIQUE exit program to indicate that UERCAKLL was returned previously to purge the queue.
- For specific modegroup allocate requests, use return code UERCAKLM. UERCAKLM also returns SYSIDERR to all application programs waiting on the purged allocate requests. CICS sets the UEPFLAG parameter to UEPRC12 on subsequent calls to your XZIQUE exit program to indicate that UERCAKLM was returned previously to purge the queue.

Purging a queue that is causing congestion in the flow of tasks frees task slots that are needed to prevent the system becoming clogged. The more you allow a session queue to grow, the more likely you are to reach the task ceiling set by the MAXT parameter, and then cause a queue of incoming tasks in the local region that cannot be attached. Note that some internal CICS requests (such as those for the LU services model transactions CLS1, CLS2, and CLS3) are not purged by return codes UERCAKLL and UERCAKLM.

If a queue has been purged previously (with UERCAKLL or UERCAKLM) but there are no queued requests currently, check the number of successful allocates since the queue was last purged. For non-specific allocate requests, this number is in UEPSARC8, and for specific modegroup requests, this number is in UEPMAR12. If no requests of this type have been allocated on this connection since the queue was last purged, the problem that caused the purge previously has not been resolved, and this request should be rejected with UERCAPUR.

If the UEPSARC8 or UEPMAR12 parameters show that allocates are being processed, you should use UERCAQUE to resume queuing of requests. If you return with UERCAQUE in this case, CICS issues an information message to the console to signal that queuing has been resumed.

Note: The address of the system entry statistics record, UEPCONST, is supplied for both non-specific and specific modegroup allocate requests.

The address of the modegroup statistics record, UEPMODST, is set to zeros for non-specific allocate requests. This address is supplied only if the request is for a specific modegroup.

If the exit is invoked after a successful allocate following the suppression of queuing, you can use the following return code:

UERCNORM
This return code indicates that CICS is to resume normal processing on the link, including queuing of requests.

Statistics fields in DFHA14DS and DFHA20DS

There are some new statistics fields maintained by CICS that your XZIQUE global user exit program can use to control queues.

A14EALRJ: Each time an XZIQUE global user exit program returns with a request to reject a request, CICS increments a new field in the system entry connection statistics. This is A14EALRJ (allocate rejected) in DSECT DFHA14DS. This field is provided to help you to tune the queue limit. Normally, if the number of sessions and the queue limit defined for a link are correctly balanced, and there has been no abnormal congestion on the link, the A14EALRJ should be zero. If the rejected allocates field is non-zero it probably indicates that some action is needed.

A14EQPCT and A20EQPCT: Each time an XZIQUE global user exit program returns with a request to purge a queue, CICS increments a new field in either the system entry or mode entry connection statistics. These fields are:

A14EQPCT The count of the number of times the queue has been purged for the connection as a whole.

A20EQPCT The count of the number of times the mode group queue has been purged.

Messages associated with intersystem queuing

There are four messages that CICS can issue in response to return codes passed back by an XZIQUE global user exit program. These are:

- DFHZC2300
- DFHZC2301
- DFHZC2309
- DFHZC2310

For details of all new, changed, and obsolete messages, see the *CICS/ESA Migration Guide*.

Exit XZIQUE

	Invoked whenever: 1. An allocate request for a session is about to be queued, or 2. An allocate request succeeds immediately following previous suppression of queuing.	
Exit-specific parameters	**UEPZDATA**	Address of the 60-byte area containing the information listed below. This area is mapped by the DSECT in copybook DFHXZIDS.
Area addressed by UEPZDATA	**UEPSYSID**	The 4-byte SYSID of the connection.
	UEPREQ	A 2-byte origin-of-request code, which can have the following values: **TR** Transaction routing **FS** Function shipping (includes distributed program link) **AL** Other kinds of intercommunication (for example, distributed transaction processing (DTP) or CPI Communications)
	UEPREQTR	The 4-byte identifier of the requesting transaction (applicable only when the origin-of-request code is FS or AL).
	UEPTRAN	The 4-byte identifier of the transaction being routed (applicable only when origin of request is TR).
	UEPFLAG	A 1-byte flag indicating whether a return code 8 or return code 12 was issued last time the exit was invoked. **UEPRC8** The exit program returned control to CICS on the previous invocation with return code 8. **UEPRC12** The exit program returned control to CICS on the previous invocation with return code 12.
	UEPPAD	A 1-byte padding field.
	UEPFSPL	Address of the 10-byte function shipping parameter list.
	UEPCONST	Address of the 80-byte system entry statistics record (this can be mapped using DSECT DFHA14DS). This parameter is applicable only to APPC connections. The modename can contain blanks.
	UEPMODST	Address of the 56-byte modegroup statistics record (this can be mapped using DSECT DFHA20DS).

Exit XZIQUE (continued)

Area addressed by UEPZDATA (continued)	**UEPSTEX**	A 6-byte area containing additional current statistics for the APPC modegroup that are not already in the modegroup statistics record (DFHA20DS), followed by a two byte padding field. The three halfword binary fields are as follows: **UEPEBND** A half-word binary field containing the number of bound sessions **UEPEWWT** A half-word binary field containing the number of contention winners with tasks **UEPELWT** A half-word binary field containing the number of contention losers with tasks **null** Two-byte padding field.
	UEPEMXQT	A half-word binary field containing the maximum queue time value specified for the connection (MAXQTIME on the CONNECTION resource definition).
Non-specific allocates data.	The following fields contain data relating to MRO, LU6.1, and non-specific APPC allocates.	
	UEPSAQTS	A double-word binary field containing the time stamp from the TCT system entry indicating the time the queue of non-specific requests was started.
	UEPSACNT	A half-word binary field containing the number of all non-specific allocates processed since the queue was started (see UEPSAQTS for the start time).
	UEPSARC8	A half-word binary field containing the number of sessions freed since the queue was last purged as a result of a UEPCAKLL return code to CICS.
Specific allocates data.	The following three fields contain data relating to specific modegroup allocates. They are applicable only when UEPMODST is non-zero (that is, it contains the address of the relevant modegroup statistics).	
	UEPMAQTS	A double-word binary field containing the time stamp from the TCT mode entry indicating the time that the modegroup queue was started for this specific modegroup.
	UEPMACNT	A half-word binary field containing the number of all specific allocates for this modegroup processed since the queue was started (see UEPMAQTS for the start time).

Exit XZIQUE (continued)

Area addressed by UEPZDATA (continued)	**UEPMAR12**	A half-word binary field containing the number of modegroup sessions freed since the queue was last purged as a result of a UEPCAKLL return code to CICS.
	UEPQUELM	A half-word binary field containing the queue limit specified for this connection (QUEUELIMIT on the CONNECTION definition).
Return codes	In the case of an allocate that is about to be queued, use one of the following:	
	UERCAQUE	Queue the allocate request.
	UERCAPUR	Reject the allocate request with SYSIDERR.
	UERCAKLL	Reject this allocate request with SYSIDERR. Purge all other queued allocate requests and send an information message to the operator console. CICS also returns SYSIDERR to all application programs waiting on the purged allocate requests
	UERCAKLM	Reject this allocate request for the modegroup and return SYSIDERR. Purge all other queued allocate requests for the modegroup specified on this allocate request and send an information message to the operator console. Retry the modegroup after an interval.
	UERCPURG	Task purged during XPI call.
	In the case of a successful allocate following the use of UERCAKLL or UERCAKLM, on a previous invocation of the exit, use one of the following:	
	UERCNORM	Resume normal operation of the link or modegroup.
	UERCPUR	Reject the allocate request with SYSIDERR.
XPI calls	All can be used.	

Designing an XZIQUE global user exit program

The functions of your XZIQUE exit should be designed:

1. To control of the number of tasks (and the amount of associated resource) that are waiting in a queue for a free intersystem session. Waiting tasks can degrade the performance of the local system.
2. To detect poor response from the receiving (remote) system and to notify the operator (or automatic operations program).
3. To cause CICS to issue a message when the link resumes normal operation.

The XZIQUE global user exit parameter list is designed to support these objectives.

Design considerations

The information passed at XZIQUE is designed to enable your XZIQUE global user exit program to:

- Avoid false diagnosis of problems on the connection by distinguishing poor response times from a complete bottleneck
- Ensure that a link resumes normal operation quickly and without operator intervention once any problem in a remote system is resolved.

Some guidance on the use of IRC/ISC statistics

CICS adds an entry for unsatisfied allocate requests to the following queues:

Non-specific (generic) allocate queue

All non-specific allocate requests are queued in this single queue. CICS makes the total number of entries in this queue available in the system entry statistics field A14ESTAQ, to which your global user exit program has access by means of the address of the system entry statistics, which is passed in UEPCONST.

Specific modegroup allocate queues

Specific allocate requests are queued in the appropriate modegroup queue—one queue for each specific modegroup name. CICS makes the total number of entries in all these queues available, as a single total, in the mode entry statistics field A20ESTAQ, to which your global user exit program has access by means of the address of the mode entry statistics, which is passed in UEPMODST.

Sample exit program design

A sample XZIQUE exit program is provided with CICS/ESA 4.1 as a base for you to design your own global user exit program. It is called DFH$XZIQ, and is supplied in the CICS410.SDFHSAMP library. The DSECT used by the sample program to map the area addressed by UEPZDATA is called DFHXZIDS, and this is supplied in the CICS410.SDFHMAC library.

As supplied, the sample exit program implements the same basic function as described for the QUEUELIMIT and MAXQTIME parameters on the connection resource definition. If the XZIQUE exit is not enabled, CICS uses these parameters to control the existence and length of the queue of allocate requests. If you enable the exit, the parameters from the connection definition are passed to your XZIQUE global user exit program, which can change the way in which these parameters are used.

The exit program also demonstrates how to control allocate requests for a particular modegroup, based on the same QUEUELIMIT and MAXQTIME parameters.

Overview of the sample exit program: The program uses the exit-specific parameters passed by CICS to determine the state of the connection, and to request the appropriate action, as follows:

1. The connection is operating normally; a queue may exist, but is of short length.

   In this case, the exit program returns with UERCAQUE to indicate that CICS is to **queue the request**.

2. The response from the partner system is slower than the rate of requests demands, and the queue length has grown to the limit specified on the

QUEUELIMIT parameter. The partner system is still operating normally, but is overloaded.

In this case, the exit program returns with UERCAPUR to indicate that CICS is to **purge the request**.

3. The queue has reached the limit specified by the QUEUELIMIT parameter, **and** requests that join the queue are expected to take longer to be satisfied than the time defined by the MAXQTIME parameter. (The estimated time for a request to complete is calculated by dividing the number of successful requests since the queue first formed by the time elapsed since it formed. These statistics are passed to the exit in the parameter list.)

   These criteria are used to determine that the connection is not operating correctly, and that continued queuing of tasks is not helpful. In this case:

   - The exit returns with UERCAKLL requesting CICS to **purge all queued** user requests from the connection. The SYSIDERR condition is returned to the application program.
   - CICS issues message DFHZC2300 to warn that a connection is not performing as expected.

4. The queue has been purged as a result of a previous invocation of the global user exit program, there are still no free sessions, and the request is about to be queued.

   In this case, the exit program returns with UERCAPUR to indicate that CICS is to **purge the request**. This also leaves the UEPRC8 flag set.

5. The queue has been purged as a result of a previous invocation of the global user exit program. A new allocate request has been received and is about to be allocated because a session has become free.

   CICS invokes the exit program to enable it to indicate that normal processing can continue.

   In this case, the exit program returns with UERCNORM to indicate that CICS is to **continue processing normally**. This also causes the UEPRC8 flag to be unset following this invocation, and CICS to issue message DFHZC2301.

The sample program also monitors the length of queues for modegroup-specific allocate requests and controls these—in the same way as the queue for the whole connection—using the QUEUELIMIT parameter and MAXQTIME parameters.

If both UEPRC8 and UEPRC12 are set, UERCNORM is required twice to resume normal operation. The UEPRC8 condition is reset first in this case.

Extensions to the sample program: The sample exit program does not attempt to control the queue length, or detect poor response for a particular modegroup differently from the whole connection. This kind of enhancement is something you might want to add to your own exit program if your applications request specific modegroups via the allocate command (or via a transaction profile) and you think it would be useful to control the modegroups individually.

You can also use more complex decisions (such as adding time delays to lessen the risk of false diagnosis) to decide when to issue the return codes that purge the queue, and allow queuing to restart.

End of Product-sensitive programming interface

Part 2. MVS sysplex exploitation

This Part describes MVS sysplex exploitation in CICS/ESA 4.1, under the following topics:

- Chapter 5, “Cross-system multiregion operation (XCF/MRO)” on page 87
- Chapter 6, “Dynamic transaction routing enhancements” on page 99
- Chapter 7, “Automatic restart management” on page 113.
- Chapter 8, “The IBM CICS Transaction Affinities Utility MVS/ESA” on page 123.

Chapter 5. Cross-system multiregion operation (XCF/MRO)

This chapter describes the enhancements to the CICS interregion communication (IRC) facility to provide cross-system multiregion operation (XCF/MRO) in CICS/ESA 4.1. It covers the following topics:

- Overview
- Benefits of cross-system multiregion operation (XCF/MRO)
- Requirements for XCF/MRO
- Changes to CICS externals
- Problem determination.

Overview

The CICS interregion communication (IRC) facility that supports CICS multiregion operation (MRO) is enhanced in CICS/ESA 4.1 to exploit the cross-system coupling facility (XCF[2]) of MVS/ESA, to provide dynamic add of connections, and to rationalize MRO security.

Cross-system multiregion operation (XCF/MRO)

XCF is part of the MVS/ESA base control program, providing high performance communication links between MVS images that are linked in a sysplex (**sys**tems com**plex**) by channel-to-channel links, ESCON channels, or coupling facility links[3]. (See "Requirements for XCF/MRO" on page 90 for more information about a sysplex.) The enhanced IRC provides an XCF access method, in addition to the existing IRC and XM methods that are currently available for CICS MRO connections, making it unnecessary to use VTAM to communicate between MVS images within the same MVS sysplex. Using XCF services, CICS regions join a single XCF group called DFHIR000. When they are successfully joined in the CICS XCF group, CICS regions select the XCF access method dynamically, overriding the access method specified on the connection resource definition. The use of the MVS cross-system coupling facility enables MRO to function ***between*** MVS images in a sysplex environment, supporting MRO functions such as:

- Transaction routing
- Function shipping
- Distributed program link (DPL)
- Distributed transaction processing (DTP)
- Asynchronous processing.

Note: XCF/MRO does not provide shared data table access between CICS regions in different MVS images. Shared data tables support operates only between CICS regions that reside in the same MVS image. Requests to access a shared data table in another MVS image are supported by function shipping only.

---

2 XCF. The MVS/ESA cross-system coupling facility that provides MVS coupling services. XCF services allow authorized programs in a multisystem environment to communicate (send and receive data) with programs in the same, or another, MVS image. Multisystem applications can use the services of XCF, including MVS components and application subsystems (such as CICS), to communicate across a sysplex. See the *MVS/ESA Planning: Sysplex Management* manual, GC28-1620, for more information about the use of XCF in a sysplex.

3 High bandwidth fiber optic links that provide the high-speed connectivity required for data sharing between a coupling facility and the central processor complexes directly attached to it.

The extension of the CICS interregion communication facility to operate across MVS images supports CICS regions (which can be at different release levels; see Table 6 on page 91) that reside in both the same and different MVS images, providing support for System/390 coupled systems within an MVS XCF sysplex.

Automatic selection of XCF/MRO

At connect time, a CICS region invokes the IXCQUERY macro to determine whether the MRO partner it is connecting to resides in the same MVS image. If the partner CICS region does reside in the same MVS image, CICS uses IRC or XM as the MRO access method, as defined in the connection definition.

If the partner resides in a different MVS image, and XCF is at the MVS/ESA 5.1 level or later, CICS uses XCF as the access method, regardless of the MRO access method (IRC or XM) defined on the connection definition.

Installing MRO connections with IRC open

The resource definition facilities for MRO are improved to allow new MRO connections to be dynamically added while CICS interregion communication is open. This improves operational flexibility, simplifying the task of the system administrator. However, you cannot modify existing links while IRC is open; as in earlier releases, you must close IRC to reinstall modified existing link definitions.

Security authorization for MRO by external security manager

CICS/ESA 4.1 customers benefit from improved MRO security provided by an interface to an external security manager (such as RACF).

The CICS internal security mechanisms, which provide bind-time security for MRO connections in earlier releases, are replaced by calls to an external security manager (ESM), using RACROUTE calls of the MVS system authorization facility (SAF) interface. In addition, CICS interregion communication uses the external security manager to check that CICS regions logging on to IRC are the regions they claim to be. This security enhancement is also available to earlier releases of CICS running on MVS/ESA 3.1.3 or later with RACF 1.9 and using the CICS/ESA 4.1 version of DFHIRP, the CICS interregion communication program.

Checking for the correct level of DFHIRP

A new test is introduced in CICS/ESA 4.1 to ensure that the interregion communication program, DFHIRP, is not at a lower release level than the CICS region attempting to use the services provided by DFHIRP. For CICS interregion communication to operate successfully, DFHIRP must be at the same, or higher, release level than the CICS regions using its services.

If a CICS/ESA 4.1 region detects that DFHIRP is at a lower level, it issues message DFHIR3799, and interregion communication fails to open.

Benefits of cross-system multiregion operation (XCF/MRO)

Cross-system MRO using XCF links offers many benefits to the multi-system user. Some of these benefits are:

- A much lower communication overhead between MVS images, providing much better performance than using VTAM ISC links to communicate between MVS systems
- Easier resource definitions for connections, and no VTAM tables to update
- Improved availability, by having alternative processors and systems ready to continue the workload of a failed MVS or a failed CICS
- Easy transfer of CICS systems between MVS images
- Improved bind-time security for MRO connections, with control transferred to the security administrator
- Improved price/performance, by coupling low-cost, rack-mounted, air-cooled processors, and being able to use MRO across this coupled environment
- Growth in small increments
- Removal of most constraints to growth through continued expansion of the sysplex.

MRO across the MVS sysplex

The main benefit of adding XCF/MRO to the CICS interregion communication facility is to provide efficient and flexible CICS-to-CICS communication in an MVS sysplex environment. By exploiting the MVS cross-system coupling facility, CICS supports MRO links between MVS images, enabling you to use transaction routing, function shipping, and distributed program link across MRO links in a sysplex environment, replacing the need to use CICS intersystem communication (ISC) links through VTAM for these functions. You can also use XCF/MRO for distributed transaction processing, provided the LU6.1 protocol is adequate for your purpose. The new MRO is designed to provide support for System/390 coupled systems within an MVS XCF sysplex, where each MVS image is running MVS/ESA 5.1 or later.

The simpler resource definition for MRO connections, and with no VTAM tables to update, makes it much easier to move CICS regions from one MVS to another. You no longer need to change the connection definitions from CICS MRO to CICS ISC (which, in any event, can be done only if CICS startup on the new MVS is a warm or cold start). Cross-system MRO is a prerequisite for cloning CICS application-owning regions (AORs) across a sysplex, and to be able to restart an AOR in any MVS image within a sysplex. Also, being able to add connections dynamically improves the availability of CICS by allowing regions to be added to the configuration without a service interruption.

Security improvements for MRO

The switch to an external security manager such as RACF for MRO security prevents the impersonation of CICS regions. It also provides consistent and simpler security administration by using the external security manager for all security checks.

Requirements for XCF/MRO

To communicate across MVS images through XCF/MRO requires the MVS images to be joined in a sysplex.

The sysplex environment for cross-system MRO

A sysplex consists of multiple MVS systems, coupled together by hardware elements and software services. In a sysplex, MVS provides a platform of basic multisystem services that multisystem applications like CICS can exploit. As an installation's workload grows, additional MVS systems can be added to the sysplex to enable the installation to meet the needs of the greater workload.

Usually, a specific function (one or more modules/routines) of the MVS application subsystem (such as CICS) is joined as a **member** (a member resides on one system in the sysplex), and a set of related members is the **group** (a group can span one or more of the systems in the sysplex). A group is a complete logical entity in the sysplex. To use XCF to communicate in a sysplex, each CICS region joins an XCF group as a member, using services provided by the CICS/ESA 4.1 version of DFHIRP. The group name for CICS is always DFHIR000, and the member name is always the CICS generic APPLID (NETNAME on the CONNECTION resource definition) used for MRO partners. Within the sysplex, CICS regions can communicate only with members of the CICS XCF group (DFHIR000). To join the XCF group DFHIR000, CICS regions (through DFHIRP) invoke the IXCJOIN macro when logging on to DFHIRP.

Unique generic APPLID

The CICS generic APPLID must be unique within a sysplex, even if the level of XCF is earlier than MVS/ESA 5.1, which is the minimum level for XCF/MRO support. This is because CICS, through DFHIRP, always issues the IXCJOIN macro to join the CICS XCF group when IRC is opened, regardless of the level of XCF in the MVS image. This fails with an error if the APPLID used for the XCF member name is not unique.

XCF/MRO members limit

Each XCF group has a maximum limit of 511 members, which is therefore the maximum number of CICS regions that can participate in XCF/MRO in a single sysplex. However, this limit can be reduced by:

- The number of external CICS interface users in the sysplex, because these can also be members of the CICS XCF group when using MRO links across MVS images.
- The user-defined limit for XCF groups defined in the MVS XCF couple data set. This is specified on the MAXMEMBER parameter of the DEFINEDS statement for defining the XCF couple data set. You should ensure this parameter does not restrict the number of regions that can participate in XCF/MRO.

Sysplex hardware and software requirements

The multiple MVS systems that comprise a sysplex can run in either:

- One CPC[4] (the CPC being an ESA/390-capable processing system) partitioned into one or more logical partitions (LPARs) using the PR/SM facility, or
- One or more CPCs (possibly of different processor models), with each CPC running a single MVS image, or
- A mixture of LPARs and separate CPCs.

Note: In a multi-CPC sysplex, the processing systems are usually in the same machine room, but they can also reside in different locations if the distances involved are within the limits specified for communication with the external time reference facility.

To create a sysplex that supports XCF/MRO you require:

- **MVS/ESA 5.1**—XCF is an integral part of the MVS base control program (BCP).
- **XCF coupled data sets**—XCF requires DASD data sets shared by all systems in the sysplex.
- **Channel-to-channel links, ESCON channels or high-speed coupling facility links**—for XCF signaling.
- **External time reference (ETR) facility**—when the sysplex consists of multiple MVS systems running on two or more CPCs, XCF requires that the CPCs be connected to the same ETR facility. XCF uses the synchronized time stamp that the ETR provides for monitoring and sequencing events within the sysplex.

You can run CICS/ESA 4.1 on a release of MVS earlier than MVS/ESA 5.1, but without XCF/MRO. Table 6 shows what function is available with the various levels of software.

Table 6. Release levels of XCF and DFHIRP. The minimum required level of each component to use the appropriate support.

Function	Required level of DFHIRP	Required software	Versions of CICS supported
XCF/MRO	CICS/ESA 4.1	MVS/ESA 5.1 RACF 1.9	CICS/MVS Version 2 CICS/ESA Version 3 CICS/ESA Version 4
MRO security	CICS/ESA 4.1	MVS/ESA SP 3.1.3 RACF 1.9	CICS/MVS Version 2 CICS/ESA Version 3 CICS/ESA Version 4
Dynamic install of connections	CICS/ESA 4.1	MVS/ESA SP 3.1.3 RACF 1.9	CICS/ESA 4.1

[4] CPC. One physical processing system, such as the whole of an ES/9000 9021 Model 820, or one physical partition of such a machine. A physical processing system consists of main storage, and one or more central processing units (CPUs), time-of-day (TOD) clocks, and channels, which are in a single configuration. A CPC also includes channel subsystems, service processors, and expanded storage, where installed.

Some MRO scenarios in an XCF sysplex environment

CICS MRO in an XCF sysplex environment is illustrated in Figure 6.

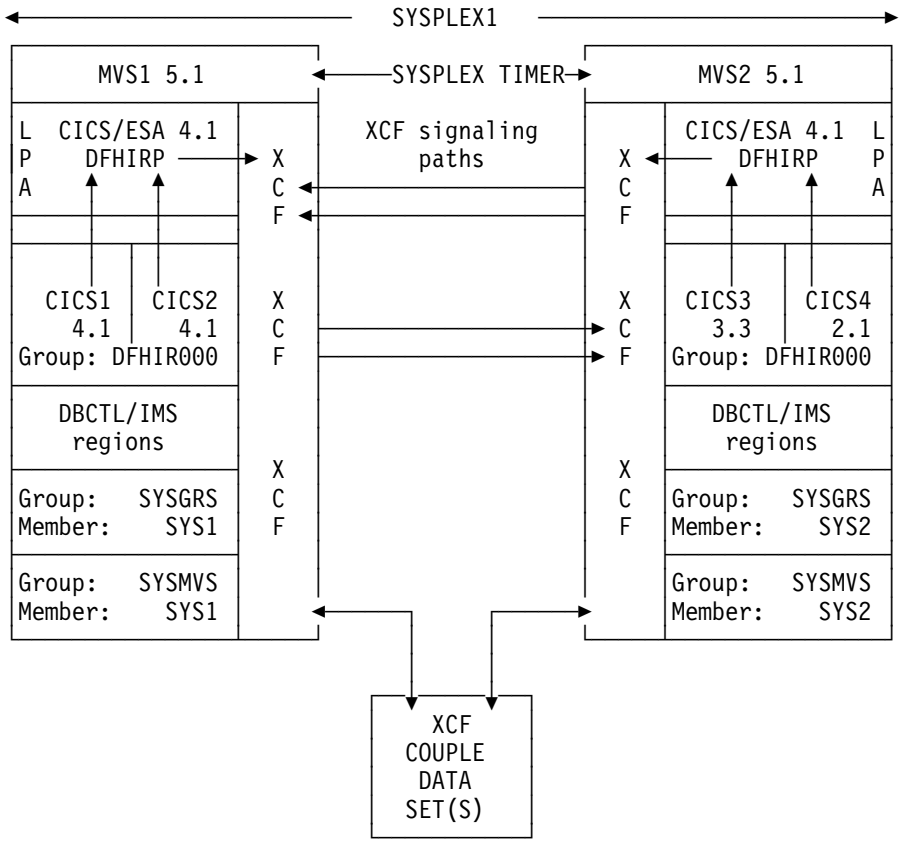


Figure 6. A sysplex (SYSPLEX1) comprising two MVS images (MVS1 and MVS2). In this illustration, the members of the CICS group, DFHIR000, are capable of communicating via XCF/MRO links across the MVS images. The CICS regions can be at the CICS/ESA 4.1 level or earlier, but DFHIRP in the LPA of each MVS must be at the CICS/ESA 4.1 level. Both MVS systems must be MVS/ESA 5.1 or later.

In Figure 6, the MRO links between CICS1 and CICS2, and between CICS3 and CICS4, use either the IRC or XM access methods, as defined for the link. The MRO links between CICS regions on MVS1 and the CICS regions on MVS2 use the XCF method, which is selected by CICS dynamically.

In a migration or coexistence situation, you may want to run with different levels of DFHIRP until you have fully tested the new release, and are ready to commit all regions to run under the CICS/ESA 4.1 version. To do this means reserving different MVS images in the sysplex for the different releases of CICS— a form of partitioning that permits different levels of DFHIRP to be used, as shown in Figure 7 on page 93.

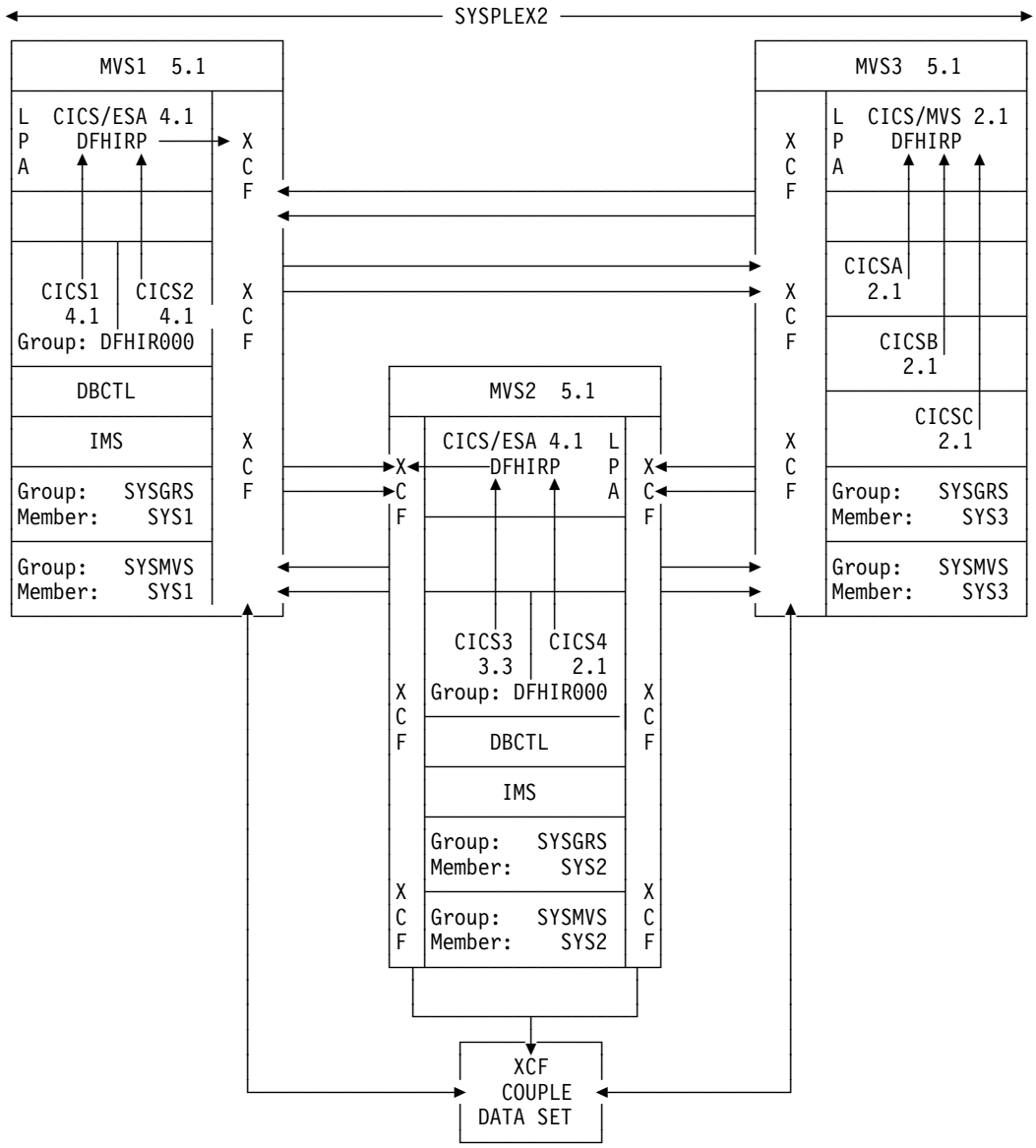


Figure 7. This illustrates a sysplex (SYSPLEX2) comprising three MVS images (MVS1, MVS2, and MVS3). The members of the CICS XCF group (DFHIR000) in MVS1 and MVS2 can communicate with each other via XCF/MRO links across the MVS images. The CICS regions in MVS3 are restricted to using MRO within MVS3 only, because DFHIRP is at the CICS/MVS 2.1 level, and cannot communicate via XCF. MVS1 and MVS2 must be MVS/ESA 5.1 or later. MVS3 can be any MVS release that includes XCF support.

Note that, in Figure 7:

- MVS3 is a member of SYSPLEX2, as shown by its XCF links with the other 2 MVS images, but it is used solely for CICS/MVS 2.1 MRO regions using the CICS 2.1 DFHIRP, which cannot use XCF. Therefore, the regions in MVS3 cannot exploit XCF to communicate across MRO links with the other CICS regions that reside in MVS1 and MVS2.
- MVS1 and MVS2 have the CICS/ESA 4.1 DFHIRP installed, and all the CICS regions in these MVS images can communicate across MRO links. The CICS regions in these MVS systems can be at the CICS Version 2, Version 3, or Version 4 level.

Changes to CICS externals

The changes to CICS IRC result in a number of changes to CICS externals, affecting the following:

- Changes to resource definition
- Changes to system definition
- Changes to user-replaceable modules

Changes to resource definition

The resource definition facilities for MRO are improved to allow new MRO connections to be dynamically added while CICS interregion communication is open.

Installing new CONNECTION definitions

There is a change to the operation of the CEDA INSTALL command for CONNECTION definitions. It is no longer necessary to close interregion communication (IRC) in order to install ***new*** connections.

Existing links, however, cannot be modified while IRC is open; as in earlier releases, IRC must be closed in order to reinstall modified link definitions. Also, the restriction that requires connections to be installed by group remains, with the install 'commit' continuing to operate at the group level. These restrictions mean that you must put any new connections in a group of their own if you want to install them while IRC is open.

SECURITYNAME parameter obsolete for MRO connection definitions

Although there are no changes to resource definition parameters, the description and function of the SECURITYNAME attribute of the CONNECTION definition is changed as a result of the security changes for CICS MRO bind-time security.

The SECURITYNAME is no longer used for MRO bind-time security checking, nor is it used for any other security purpose on MRO links. Bind-time security is performed by calls to an external security manager via the MVS SAF interface. Note that this is also true for earlier releases of CICS that are communicating through the CICS/ESA 4.1 version of DFHIRP.

See "Signon for link security" on page 96 for information about link security.

Changes to system definition

There are no changes to CICS system definitions, but the CICS internal security mechanisms that provide bind-time security for MRO connections are replaced by calls to an external security manager, using RACROUTE calls of the MVS system authorization facility (SAF) interface. In addition, CICS interregion communication uses the external security manager to check that CICS regions logging on to IRC are not impostors. These changes in MRO security require the definition of profiles and authorizations for CICS APPLIDs in the RACF FACILITY class.

Note: If you are using the CICS shared data tables feature, the RACF profiles you require for MRO security are identical to those required for shared data tables security. These are described under "Security definitions" on page 95.

Security definitions

The change from CICS internal security to management of authentication and bind-security by an external security manager requires the definition of new profiles in the RACF database. The ESM implementation of bind-time security replaces the call to DFHACEE. There are two security checks in the CICS/ESA 4.1 DFHIRP:

- At logon time
- At connect time

These security checks, via RACROUTE calls to the SAF interface, are always performed, regardless of whether the MRO partner regions are running with external security active for CICS resource security checking.

Logon security checking: To enforce MRO logon security, each region that logs on to DFHIRP must be authorized to RACF in a DFHAPPL.*applid* profile in the RACF FACILITY class. This requires the definition of a DFHAPPL.*applid* profile for each region that logs on to DFHIRP, and that each CICS region userid has UPDATE access to its own DFHAPPL.*applid* profile. This form of authorization enables DFHIRP to check that each CICS region attempting to log on is the region it claims to be, and not an imposter. See Figure 8 for an illustration of logon checking.

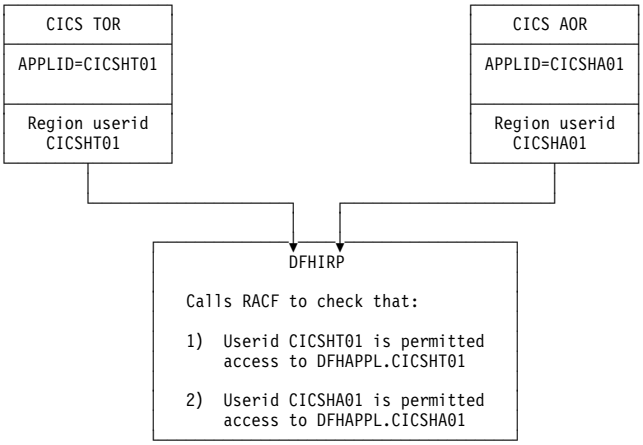


Figure 8. Illustration of the DFHIRP logon security check

The terminal-owning region (TOR) and application-owning region (AOR) shown in Figure 8, running under userids CICSHT01 and CICSHA01 respectively, with APPLIDs CICSHT01 and CICSHA01, require the following RACF definitions to authorize their logon to DFHIRP:

```
RDEFINE FACILITY (DFHAPPL.CICSHT01) UACC(NONE)
RDEFINE FACILITY (DFHAPPL.CICSHA01) UACC(NONE)

PERMIT DFHAPPL.CICSHT01  CLASS(FACILITY) ID(CICSHT01) ACCESS(UPDATE)
PERMIT DFHAPPL.CICSHA01  CLASS(FACILITY) ID(CICSHA01) ACCESS(UPDATE)
```

These RACF definitions authorize each region to log on to DFHIRP, and verifies that they are the regions they claim to be.

Connect (bind-time) security: To enforce MRO connect (bind-time) security, each region that connects to another region via MRO must be authorized to RACF in a DFHAPPL.*applid* profile in the RACF FACILITY class. This means that each CICS region in an MRO interregion communication link must be given access to its

partner's DFHAPPL.*applid* profile, with READ authority required for connect security. For example, for the CICS TOR running under userid CICSHT01 (with APPLID CICSHT01), that connects to the AOR running under userid CICSHA01 (with APPLID CICSHA01), use the following RACF commands to authorize the connections:

```
PERMIT DFHAPPL.CICSHT01  CLASS(FACILITY) ID(CICSHA01) ACCESS(READ)
PERMIT DFHAPPL.CICSHA01  CLASS(FACILITY) ID(CICSHT01) ACCESS(READ)
```

This example does not show the associated RDEFINEs for the DFHAPPL.*applid* profiles, because these must already exist for logon security checking.

You cannot specify to CICS whether or not you want connect security checking for MRO connections—CICS always issues the RACROUTE calls. The SECURITYNAME option on connection resource definitions is applicable to ISC links only, and cannot be used to control bind security for MRO. The SECURITYNAME parameter is also redundant for MRO *link* security. See “Signon for link security” for information about changes affecting MRO link security.

If you want to avoid defining and maintaining specific authorizations, you can:

- Specify the UACC(READ) option on the FACILITY resource class profiles, or
- Omit the FACILITY class profiles altogether.

If SAF neither grants nor refuses an access request: If the security profile for a specified resource is not retrieved, SAF neither grants nor refuses the access request. In this situation:

- IRC rejects the logon or connection request if a security manager is installed, but is either temporarily inactive or inoperative for the duration of the MVS image. This is a fail-safe action, on the grounds that, were the security manager active, it might retrieve a profile that does not permit access.
- IRC allows the logon or connection if:
  - There is no security manager installed, or
  - There is an active security manager, but the FACILITY resource class is undefined or inactive, or
  - There is no profile in the FACILITY class installed in the CICS region that initiated the call.

  The logon is allowed in these cases, as there is no evidence that you want to control access to the CICS APPLID.

CICS determines whether a security manager is present or not using SAF return codes.

Signon for link security

The SECURITYNAME parameter on the connection resource definition is ignored for link security on MRO links, and is supported for APPC and LU6.1 links only.

For MRO link security, CICS signs on sessions using one of the following:

1. The RACF region userid passed by its partner at MRO logon—the same userid that is used for bind-time security.
2. When specified, the USERID parameter on the SESSIONS resource definition.

If specified, a USERID on the SESSIONS definition overrides the RACF region userid.

Changes to user-replaceable modules

The user-replaceable security identification program, DFHACEE, which was previously used to obtain the CICS region's userid, is withdrawn in CICS/ESA 4.1. Instead, the CICS interregion communication program, DFHIRP, obtains its caller's userid directly from the appropriate accessor environment element (ACEE). It is thus not possible to “customize” the CICS region userid as you can in earlier releases using DFHACEE. For MRO purposes, this change also affects any earlier releases of CICS that are using MRO with the CICS/ESA 4.1 interregion communication program (DFHIRP) because DFHACEE is no longer called from DFHIRP. **However, DFHACEE is also used during CICS security initialization in earlier releases of CICS, if your RACF profiles are prefixed with the CICS region userid (that is, the system initialization parameter SECPRFX=YES is specified). Therefore, you should ensure that an appropriate version of DFHACEE is maintained in the linklist for use during security initialization in your earlier CICS releases.**

The system authorization facility (SAF) and the SAF router are present on all MVS systems, even if RACF is not installed. Although the SAF router is not part of RACF, many system components and programs, such as CICS, invoke RACF through the RACROUTE macro and SAF. Therefore, installations can modify RACF parameter lists and do customized security processing within the SAF router. For information about how to code a SAF router exit, see the *External Security Interface (RACROUTE) Macro Reference for MVS and VM* manual, GC28-1366.

Problem determination

There are new messages and trace entries in CICS/ESA 4.1 to aid problem determination for XCF/MRO.

New messages

For details of all the new, changed, and obsolete messages, see the *CICS/ESA Migration Guide*.

Trace

Problem determination in the interregion communication area is more complex with the addition of the XCF access method to the existing IRC and XM methods, particularly when the communicating regions are running in different MVS systems.

There are new trace entries for MRO in CICS/ESA 4.1, and these are described in the *CICS/ESA Diagnosis Reference*.

You can also diagnose problems in the IRC area by using:

- MVS traces to follow the DFHIRP and XCF flows
- CICS dumps of the MRO control blocks
- Internal CICS trace of the DFHIR calls

CICS MRO trace entries trace R0 from the DFHIR call, which in the case of RACF is the RACF return code.

The security return code IRERRACE-204 includes the RACF return code in R0 which is included in the CICS trace following the DFHIR call. A message is issued when a region attempts to logon to DFHIRP with an APPLID that it is not authorized to use.

DFHIRP runs in supervisor state, but most of the other changed code is intended to run unauthorized. The MULTIPART and MSGSTGKEY key words provided by MVS/ESA 5.1 IXCMSGO and IXCMSGI are used to read or write the user data in a different storage key from the control data.

For more information, see the *CICS/ESA Problem Determination Guide*.

Storage usage

The storage usage for MRO is changed in CICS/ESA 4.1. Some control blocks are moved from the CSA into the private area of the CICS address space, and there is some increase in the size of MRO control blocks generally. When using MRO/XCF, XCF uses its own storage buffers.

Chapter 6. Dynamic transaction routing enhancements

This chapter describes the enhancements to dynamic transaction routing in CICS/ESA 4.1. It covers the following topics:

- Overview
- Benefits of dynamic transaction routing enhancements
- Requirements for dynamic transaction routing enhancements
- Changes to CICS externals
- Problem determination.

Overview

The CICS dynamic transaction routing mechanism is enhanced to allow a dynamic transaction routing program to maintain better information about the state of routed transactions. This enables such a program to make more intelligent routing decisions.

The CICS relay program, DFHCRP, is replaced by a new relay program, DFHAPRT. The new relay program invokes the user-replaceable dynamic transaction routing program at several new points, as follows:

- Before routing a remote, terminal-oriented, transaction that is initiated by automatic transaction initiation (ATI)
- At the end of a transaction that DFHAPRT has routed locally, if the initial invocation requests re-invocation at termination
- If a routed transaction abends, to indicate that an abend occurred before returning a response to the CICS transaction manager.

The DFHAPRT program passes information to the dynamic transaction routing program by means of a communication area (COMMAREA). The COMMAREA has some new function codes and new fields, as described in “Parameters passed to the dynamic transaction routing program” on page 103.

Benefits of dynamic transaction routing enhancements

The enhancements to the dynamic transaction routing mechanism offer a number of benefits to user of dynamic transaction routing:

- They enable a dynamic transaction routing program to make more intelligent routing decisions; for example, based on workload goals.
- They enable CICS to provide improved support for MVS goal-oriented workload management.
- They make it easier to use a global temporary storage owning region in the MVS sysplex environment, which avoids intertransaction affinity that can occur with the use of local temporary storage queues.
- They are used by CICSPlex SM to provide intelligent routing in a CICSplex that has at least one terminal-owning region linked to multiple application-owning regions.
- They make it possible to detect intertransaction affinities as they are created.

Requirements for dynamic transaction routing enhancements

There are no special requirements to be able to use the dynamic transaction routing enhancements introduced in CICS/ESA 4.1, but you should consider the additional advantages from using CICSPlex SM.

For information about CICSPlex SM, see the *CICSPlex SM: Concepts and Planning* manual, GC33-0786.

Changes to the dynamic transaction routing user-replaceable program

Product-sensitive programming interface

A user dynamic transaction routing program is invoked for additional reasons by the CICS transaction relay program, DFHAPRT, in CICS/ESA 4.1.

CICS invokes the user-replaceable dynamic 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/ESA 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 reinvocation at termination.
- If a routed transaction abends, if the initial invocation requests re-invocation at termination.

Information passed to the dynamic transaction routing program

The CICS relay program, DFHAPRT, passes information to the dynamic transaction routing program by means of a communications area. The communications area contains fields that are mapped by the DSECT DFHDYPDS, and is described in detail in “Parameters passed to the dynamic transaction routing program” on page 103. Some of the data passed to the dynamic transaction routing program in the communications area is:

- The SYSID of the remote CICS region specified when the transaction was installed
- The netname of the remote CICS region
- The name of the remote transaction
- The dispatch priority (MRO only) of the remote transaction
- Whether or not the request is to be queued if no sessions are immediately available to the remote CICS region
- The address of the remote transaction’s communications area
- The address of a copy of the transaction’s terminal input/output area (TIOA)
- A task-local user data area.

The communications area DSECT contains comments to describe the information passed.

The dynamic transaction routing program can accept these values, or change them, or tell CICS not to continue routing the transaction. The values used depend on the function being performed; that is, some values may be ignored.

The information passed to the dynamic transaction routing program indicates whether the transaction is being routed dynamically or statically (for remote ATI requests).

If the transaction is being routed dynamically, the dynamic transaction routing program can change the SYSID or netname to determine where the transaction is to run.

If the transaction was started by ATI, the dynamic transaction routing program is called only to notify it of where the transaction is going to run. In this case, the dynamic transaction routing program cannot change the remote system name, and any changes to the SYSID or NETNAME fields in the communications area are ignored.

For transactions that are run remotely, either because they are defined as remote or because they are dynamically routed to a remote CICS region, CICS monitoring is informed of the SYSID of the remote CICS region. For transactions that the dynamic transaction routing program routes locally, the monitoring field is set to nulls.

Changing the target CICS region

The communications area passed to the dynamic transaction routing program initially contains the SYSID and netname of the default CICS region to which the transaction is to be routed. These are derived from the value of the REMOTESYSTEM option of the installed transaction definition. If the transaction definition does not specify a REMOTESYSTEM value, the SYSID and netname passed are those of the local CICS region.

Note: The recommended method is to use a single, common definition for all remote transactions that are to be dynamically routed. The name of the common definition is specified on the DTRTRAN system initialization parameter. For information about defining remote transactions for dynamic transaction routing, see the *CICS/ESA Intercommunication Guide*.

The dynamic transaction routing program can change the SYSID and netname, so that the CICS region to which the transaction is routed is determined as follows:

- The NETNAME and the SYSID are not changed.

  CICS tries to route to the SYSID as originally specified in the communications area.

- The NETNAME is not changed, but the SYSID is changed.

  CICS updates the communications area with the NETNAME corresponding to the new SYSID, and tries to route to the new SYSID.

- The NETNAME is changed, but the SYSID is not changed.

  CICS updates the communications area with a SYSID corresponding to the new NETNAME, and tries to route to the new SYSID.

- The NETNAME is changed **and** the SYSID is changed.

  CICS overwrites the communications area with a SYSID corresponding to the new NETNAME, and tries to route to that new SYSID.

If the NETNAME specified is invalid, or cannot be found, SYSIDERR is returned to the dynamic transaction routing program.

Changing the program name

The DYRLPROG field in the communications area specifies an alternative program name that can be used if the transaction is routed locally. For example, if all remote CICS regions are unavailable and the transaction cannot be routed, you may want to run a program in the local CICS terminal-owning region to send an appropriate message to the user.

Telling CICS whether to route or terminate a transaction

If you want a transaction to be routed, whether you have changed any values or not, you return a zero value to CICS in field DYRRETC of the communications area. If you want to terminate the transaction with a message or an abend, you supply a return code of `X'8'`. A further option (return code `X'4'`) tells CICS to terminate the transaction without issuing a message or abend.

Warning: Setting a return code of `X'4'` for APPC transaction routing leads to unpredictable results, and should be avoided.

When you return control to CICS with return code zero, CICS first compares the returned SYSID with its own local SYSID:

- If the SYSIDs are the same (or the returned SYSID is blank) CICS executes the transaction locally.
- If the two SYSIDs are not the same, CICS routes the transaction to the remote CICS region, using the remote transaction name.

The dynamic transaction routing program is invoked again if the routed transaction abends and, optionally, when the routed transaction terminates.

Invoking the dynamic transaction routing program at end of routed transactions

If you want your dynamic transaction routing program to be invoked again when the routed transaction has completed, you must set the DYROPTER field in the communications area to `'Y'` before returning control to CICS. You might want to do this, for example, if you are keeping a count of the number of transactions currently executing on a particular CICS region. However, during this reinvocation, the dynamic transaction routing program should update only its own resources, because at this stage the final command to the terminal from the application program in the AOR may be pending, and the dynamic transaction routing program is about to terminate.

Note: If you have set DYROPTER to `'Y'`, and the routed transaction abends, the dynamic transaction routing program is invoked again to notify it of the abend. You could use this invocation to initiate a user-defined program in response to the transaction abend.

Modifying the application's communications area

Sometimes you may want to modify the routed application's communications area. For example, if your routing program changes the ID of the remote transaction, it may also need to change the input communications area passed to the routed application. Field DYRACMAA of the routing program's communications area enables you to do this; it is a pointer to the application's communications area.

Receiving information from a routed transaction

If your dynamic transaction routing program chooses to be reinvoked at the end of a routed transaction, it can obtain information about the transaction by monitoring its output communications area and output TIOA.

Monitoring the output communications area

A routed transaction can pass information back to the dynamic transaction routing program in its output communications area. When invoked at transaction termination, your routing program can examine the output communications area (pointed to by DYRACMAA). The following is an example of how this facility could be used:

You have a CICSplex consisting of sets of functionally-equivalent TORs and AORs, and need to identify any intertransaction affinities that may affect transaction routing. You could use the CICS Transaction Affinities Utility[5] to do this, but there are some affinities that the utility cannot detect (for example, those created by non-CICS functions). Also, some transactions may sometimes create affinities, and sometimes not.

However, the routed transactions themselves "know" when an affinity is created, and can communicate this to the dynamic transaction routing program. The routing program is then able to route such transactions accordingly.

Monitoring the output TIOA

When invoked at transaction termination, your routing program can examine the copy of the routed transaction's output TIOA pointed to by DYRBPNTR. This can be useful, for example, to guard against the situation where one AOR in a CICSplex develops software problems. These may be reported by means of a message to the end user, rather than by a transaction abend. If this happens, the routing program is unaware of the failure and cannot bypass the AOR that has the problem. By reading the output TIOA, your routing program can check for messages indicating specific kinds of failure, and bypass any AOR that is affected.

Parameters passed to the dynamic transaction routing program

Figure 9 on page 104 shows all the parameters passed from DFHAPRT, the CICS relay program, to the dynamic transaction routing program by means of a communications area. The communications area is mapped by the copy book DFHDYPDS, which is in the appropriate CICS library for all the supported programming languages.

[5] The IBM CICS Transaction Affinities Utility MVS/ESA, program number 5696-582.

```
                DS      OCL4        Standard Header
DYRFUNC         DS      CL1         Function Code
DYRCOMP         DS      CL2         Component Code         Always 'RT'
DYRFILL1        DS      CL1         Reserved
DYRERROR        DS      CL1         Route selection error code
DYROPTER        DS      CL1         Transaction termination option
DYRQUEUE        DS      CL1         Queue-the-request indicator
DYRFILL2        DS      CL1         Reserved
DYRRETC         DS      F           Return code
DYRSYSID        DS      CL4         Default/Selected sysid
DYRVER          DS      CL4         Version of the interface
DYRFILL3        DS      CL4         Reserved
DYRTRAN         DS      CL8         Default/Selected remote tranid
DYRCOUNT        DS      F           Number of invocations count
DYRBPNTR        DS      F           Address of input buffer
DYRBLGTH        DS      F           Length of input buffer
DYRRTPRI        DS      CL1         Route priority to AOR
DYRFILL4        DS      CL1         Reserved
DYRPRTY         DS      H           Dispatch priority on the TOR
DYRNETNM        DS      CL8         Netname matching sysid
DYRLPROG        DS      CL8         Run this program if routed locally
DYRDTRXN        DS      CL1         DTRTRAN indicator
DYRDTRRJ        DS      CL1         DTRTRAN reject
DYRSRCTK        DS      XL4         MVS WLM source token
DYRABCDE        DS      CL4         Transaction abend code
DYRCABP         DS      CL1         Continue abend processing?
DYRFILL5        DS      CL4         Reserved
DYRACMAA        DS      F           Address of applications's commarea
DYRACMAL        DS      F           Length of application's commarea
DYRUSER         DS      CL128       User area
```

Figure 9. The communication area passed to a dynamic transaction routing program

DYRABCDE
is the abend code returned when a remote or locally routed transaction abends.

DYRACMAA
is the 31-bit address of the routed application's communications area. If there is no communications area, this field is set to null.

When your dynamic routing program is invoked for routing (DYRFUNC=0), the address is that of the input communications area (if any). Likewise, when your routing program is invoked because of a route-selection error or a remote ATI request (DYRFUNC=1 and 3, respectively), the address is that of the input communications area.

When your routing program is invoked because a previously-routed transaction has terminated normally (DYRFUNC=2), the address is that of the output communications area (if any). Routed applications can use their output communications area to pass information to the dynamic routing program—see "Receiving information from a routed transaction" on page 103.

When your routing program is invoked because the routed transaction has abended (DYRFUNC=4), the information in the communications area is not meaningful.

Your routing program can alter the data in the application's communications area.

DYRACMAL
is the length of the routed-to application's communications area. If there is no communications area, this field is set to zero.

DYRBLGTH
is the length of the copy of the TIOA/LUC buffer.

DYRBPNTR
is the 31-bit address of a copy of the TIOA/LUC buffer.

When your dynamic routing program is invoked for routing, because of a route-selection error, or for a remote ATI request (DYRFUNC=0, 1, and 3, respectively), it is given a copy of the input TIOA. Your routing program can alter the terminal input data passed to the routed transaction—see the *CICS/ESA Customization Guide* for information about modifying input/output areas.

When your routing program is invoked because a previously-routed transaction has terminated normally (DYRFUNC=2), it is given a copy of the output TIOA. Your routing program can monitor the output TIOA to detect possible problems in the AOR—see "Receiving information from a routed transaction" on page 103.

DYRCABP
indicates whether or not you want CICS to continue standard abend processing. The possible values are:

Y Continue with CICS abend processing.

N Terminate the transaction, do not continue with CICS abend processing, and give control to the program specified by DYRLPROG.

This option enables you to pass control to a local program that can handle the condition in your own way, and issue appropriate messages to terminal users.

If you enter N, you must ensure that DYRLPROG specifies the name of a valid program on the local system.

There is no default value.

DYRCOUNT
is a count of the times the dynamic transaction routing program has been invoked for routing purposes for this transaction with DYRFUNC set to '0', '1', or '3'. This option is provided to allow you to limit the number of retry requests.

DYRDTRRJ
indicates whether the transaction, which is defined by the common transaction definition specified on the DTRTRAN system initialization parameter, is to be rejected, or accepted for processing. This parameter is only relevant when DYRTRXN is set to Y. The possible values are:

Y The transaction is rejected. This is the default.

N The transaction is not rejected.

This indicator is always set to the reject condition when the dynamic transaction routing program is invoked. To dynamically route a transaction defined by the DTRTRAN definition, you must change this to the accept condition.

If you reject the transaction, message DFHAC2001—"Transaction *'tranid'* is unrecognized"—is sent to the user's terminal.

DYRDTRXN
indicates whether the transaction to be routed is defined by the common transaction definition specified on the DTRTRAN system initialization parameter, or by a specific transaction definition. The possible values are:

Y The transaction is defined by the definition specified by the system initialization parameter DTRTRAN. That is, there is no resource definition for the input transaction identifier (id).

The transaction is initiated in the terminal-owning region using the transaction id specified by the system initialization parameter, DTRTRAN. The input transaction id is passed to the dynamic transaction routing program in the DYRTRAN field.

N The transaction is not defined by the definition specified by the system initialization parameter, DTRTRAN. There is an installed resource definition for the input transaction id.

The transaction is initiated in the terminal-owning region using the input transaction id. The transaction id passed to the dynamic transaction routing program in the DYRTRAN field is the remote transaction id from the transaction resource definition (if this is different from the input transaction id).

For an explanation of the DTRTRAN system initialization parameter, see the *CICS/ESA System Definition Guide*.

DYRERROR
has a value only when DYRFUNC is set to '1'. It indicates the type of error that occurred during the last attempt to route a transaction. The possible values are:

0 The selected sysid is unknown.

1 The selected system is not in service.

2 The selected system is in service, but no sessions are available.

3 An allocate request has been rejected, and SYSIDERR returned to the application program. This error occurs for one of the following reasons:

1. An XZIQUE global user exit program requested that the allocate be rejected, or
2. CICS rejected the allocate request automatically because the QUEUELIMIT value specified on the CONNECTION resource definition has been reached.

4 A queue of allocate requests has been purged, and SYSIDERR returned to all the waiting application programs. This error occurs for one of the following reasons:

1. An XZIQUE global user exit program requested that the queue be purged, or
2. CICS purged the queue automatically because the MAXQTIME limit specified on the CONNECTION resource definition has been reached.

DYRFUNC
tells you the reason for this invocation of the dynamic transaction routing program. The possible values are:

0 Invoked for transaction routing

1 Invoked because an error occurred in route selection

2 Invoked because a previously routed transaction has terminated

3 Invoked before automatic initiation

4 Invoked because the routed transaction abended.

DYRLPROG
is the name of the initial program associated with the transaction for which the dynamic transaction routing program is invoked, if the transaction is defined for dynamic routing. If the transaction is being statically routed (it is defined with DYNAMIC(NO) and a specific remote system name), this field contains blanks.

You can use this field to specify the name of an alternative program to be run if the transaction is routed locally. For example, if all remote CICS regions are unavailable, and the transaction cannot be routed, you may want to run a program in the local terminal-owning region to send an appropriate message to the user.

Note: DYRLPROG must not be set to blanks when you specify DYRCABP=N. If you specify DYRCABP, ensure you also specify a valid program name on DYRLPROG.

DYRNETNM
is the netname of the CICS region identified in DYRSYSID.

If the DYRNETNM value is changed by the initial invocation of the dynamic transaction routing program, CICS tries to route the transaction to the CICS region with the new netname.

DYROPTER
specifies whether the dynamic transaction routing program is to be reinvoked when the routed transaction terminates. The possible values are:

N The dynamic transaction routing program is not to be reinvoked. This is the default.

Y The dynamic transaction routing program is to be reinvoked.

You can specify this option for transactions that are routed to remote CICS regions and also for transactions that are executed locally.

DYRQUEUE
identifies whether or not the request is to be queued if no sessions are immediately available to the remote system identified by DYRSYSID. The possible values are:

Y The request is to be queued if necessary. This is the default.
N The request is not to be queued.

DYRPRTY
can be used to set the dispatch priority of the task in the application-owning region, if the connection between the terminal-owning region and application-owning region is MRO.

CICS sets this value to '0' (zero) before invoking the dynamic transaction routing program. If the DYRRTPRI value is 'Y' on return from the initial invocation, CICS passes the DYPPRTY value to the application-owning region.

DYRRETC
contains a return code that tells CICS how to proceed. The possible values are:

0 Continue processing the transaction.
4 Terminate the transaction without message or abend.
8 Terminate the transaction with either a message or an abend.

Whenever the routing program is invoked, DYRRETC is set to '0'. If you want CICS to continue processing the transaction, you must leave it set to '0'.

To make CICS terminate the transaction (issuing a message or abend), return a value of '8'.

To make CICS terminate the transaction without issuing a message or abend (indicating that DFHDYP has done all the processing that is necessary), return a value of '4'.

Warning: Setting a return code of '4' for APPC transaction routing leads to unpredictable results, and should be avoided.

You do not need to set a return code when the routing program is invoked at transaction termination. (Any code you set is ignored by CICS.)

DYRRTPRI
indicates whether or not the dispatch priority of the transaction should be passed to the application-owning region, if the connection between the terminal-owning region and the application-owning region is MRO. The possible values are:

N The dispatch priority is not passed. This is the default.

Y The dispatch priority is passed.

DYRSRCTK
is the MVS workload management service and reporting class token for the routed transaction.

DYRSYSID
identifies the SYSID of a CICS region. The exact meaning of this parameter depends on the value for DYRFUNC:

- When DYRFUNC is set to '0', DYRSYSID contains:
  - The remote CICS region name as specified on the REMOTESYSTEM option of the installed transaction definition, or,
  - The system name of the local CICS region if REMOTESYSTEMNAME is not specified.

  The dynamic transaction routing program can accept this value or change it before returning to CICS.

  If the SYSID you return to CICS is the same as the local sysid, CICS runs the transaction in the local region (the terminal-owning region).

- When DYRFUNC is set to ‘1’, DYRSYSID contains the CICS region name returned to CICS by the dynamic transaction routing program on its previous invocation, and the SYSID is found to be in error.

  The action your dynamic transaction routing program can take when DYRFUNC=1 depends on the DYRERROR parameter setting:

  – If DYRERROR is set to ‘0’ (unknown sysid) or ‘1’ (CICS region not in service) and you want CICS to retry transaction routing, you must change DYRSYSID before returning to CICS.

  – If DYRERROR is set to ‘2’ (no session available) and you want CICS to retry transaction, you must change DYRSYSID or change the value of DYRQUEUE to ‘Y’ (queue the request until a session is available).

- When DYRFUNC is set to ‘2’, DYRSYSID contains the name of the CICS region on which the completed transaction executed.
- When DYRFUNC is set to ‘3’, DYRSYSID contains the CICS region name specified on the REMOTESYSTEM option of the installed transaction definition, with DYNAMIC(NO) also specified. Any changes to this value, or to DYRNETNAME, are ignored.
- When DYRFUNC is set to ‘4’, DYRSYSID contains the name of the CICS region on which the transaction abended.

DYRTRAN
contains the remote transaction id.

When DYRFUNC is set to ‘0’ or ‘3’, DYRTRAN contains the remote transaction id specified on the REMOTENAME option of the installed transaction definition. Your dynamic transaction routing program can accept this remote transaction id, or supply a different transaction name for forwarding to the remote CICS region. If the supplied name is longer than four characters, it is truncated by CICS.

You can change DYRTRAN on any call to the dynamic transaction routing program, though it is effective only when DYRFUNC is set to ‘0’, ‘1’, or ‘3’.

DYRUSER
is a 128-byte user area.

CICS initializes this user area to zeroes before invoking the dynamic transaction routing program for a given task. This user area can be modified by the dynamic transaction routing program, returned to the CICS relay program, DFHAPRT, and is passed to subsequent invocations of the dynamic transaction routing program for a given transaction instance.

DYRVER
is the version number of the dynamic transaction routing program interface. The default is “2”.

Testing your dynamic transaction routing program

You can use the CICS execution diagnostic facility (EDF) to test your dynamic transaction routing program. To do so, you must name your program something other than DFHDYP, because you cannot use EDF for programs that begin with “DFH”. For details of how to use EDF, see the *CICS/ESA Application Programming Guide*.

You can use EDF in either single- or dual-terminal mode. If you choose single-terminal mode, EDF displays screens for both the dynamic transaction routing program and the application program that is invoked by the routed transaction. The screens relate to:

- The initial invocation of the dynamic transaction routing program for transaction routing (DYRFUNC=0 or DYRFUNC=3)
- The invocation of the dynamic transaction routing programm if an error occurs in route selection (DYRFUNC=1)
- The invocation of the application program
- The termination of the task
- The invocation of the dynamic transaction routing program at termination of the routed transaction (DYRFUNC=2), if you have specified DYROPTER=Y
- The invocation of the dynamic transaction routing program if the routed transaction abends (DYRFUNC=4).

If you want EDF to display the execution of your dynamic routing program only, either choose dual-terminal mode, or use one of the other methods described in the *CICS/ESA Application Programming Guide*.

The CICS sample dynamic transaction routing program, DFHDYP

The CICS-supplied sample dynamic transaction routing program is an assembler-language command-level program, named DFHDYP. The corresponding sample copy book that defines the communication area is DFHDYPDS. In addition, there are COBOL, PL/I, and C source-level samples and copy books.

When invoked with DYRFUNC set to '0' or '3', the sample programs accept the sysid and remote transaction name that are passed in fields DYRSYSID and DYRTRAN of the communication area, and set DYRRETC to '0' before returning to CICS. When invoked with DYRFUNC set to '1' or '4', they set a return code of '8'.

If you want to route transactions dynamically, you must customize DFHDYP, or replace it completely with your own routing program.

Note: The CICSPlex SM product provides a full-function dynamic transaction routing program that uses more complex algorithms to route transactions.

End of Product-sensitive programming interface

Changes to resource definition

DFHCRP is moved from the CICS-supplied resource definition group included in DFHLIST to one of the compatibility resource groups. Also, it is no longer possible for application programs to link to DFHCRP by an EXEC CICS LINK command.

Changes to monitoring and statistics

Once CICS has established that a transaction is to be routed, CICS monitoring is provided with the related SYSID and new program name (specified on the DYRLPROG parameter).

Problem determination

There are changes to aid problem determination in connection with dynamic transaction routing in the form of new trace points.

There are also some minor changes to messages and abend codes as a result of DFHAPRT replacing DFHCRP. There is also one new abend code.

Trace

New trace entry and exit points are provided so that level one tracing causes the values of the parameters related to the DFHAPRT gate in the AP domain to be recorded on entry and exit.

In situations that it cannot handle, the CICS transaction relay program, DFHAPRT, returns an exception to CICS transaction manager, and records trace entries and a dump in the same way as DFHCRP.

Changes to messages and abend codes

There are some minor changes to messages and codes in CICS/ESA 4.1 for dynamic transaction routing, and one new abend code.

The messages affected are as follows:

DFHRT4416	DFHRT4419
DFHRT4417	DFHRT4420
DFHRT4418	

The abend codes affected are:

ACRA	ACRF
ACRB	ACRG
ACRC	ACRH
ACRD	ACRI
ACRE	ACRJ

There is a new abend code—ACRM.

For more information about changes to messages and codes, see the *CICS/ESA Migration Guide*.

Chapter 7. Automatic restart management

This chapter describes how CICS/ESA 4.1 uses the automatic restart manager (ARM) component of MVS/ESA 5.2 to increase the availability of your systems. It covers the following topics:

- Overview
- Benefits of the MVS automatic restart manager
- Requirements for the MVS automatic restart manager
- Changes to CICS externals
- Problem determination.

Overview

MVS automatic restart management is a sysplex-wide integrated automatic restart mechanism that:

- Restarts MVS subsystems in place if they abend (or if notified of a stall condition by a monitor program).
- Restarts all the elements of a workload (for example, CICS terminal-owning regions, application-owning regions, file-owning regions, DB2, and so on) on another MVS image after an MVS failure.
- Restarts a failed MVS image.

Note: CICS/ESA 4.1 reconnects to DBCTL and VTAM automatically if either of these subsystems restart after a failure. There is no dependency on the MVS automatic restart manager for this reconnection support, which is provided automatically on any CICS/ESA 4.1 region running on the minimum supported release of MVS/ESA.

You cannot use MVS automatic restart for CICS regions running with XRF—it is available only to non-XRF CICS regions.

Registering to the MVS automatic restart manager

MVS automatic restart management is available only to those MVS subsystems that register with the automatic restart manager (ARM). CICS regions register with ARM automatically as part of CICS system initialization (but only if they are running without the CICS extended recovery facility (XRF)). If a CICS region fails before it has registered for the first time with ARM, it will not be restarted. After a CICS region has registered, it is restarted by the ARM according to a predefined **policy** for the workload.

CICS ARM processing

A prime objective of CICS support for the MVS automatic restart manager (ARM) is to preserve data integrity automatically in the event of any system failure. If CICS is restarted by ARM with the same (**persistent**) JCL, CICS forces START=AUTO to ensure data integrity.

During initialization, CICS checks with MVS to determine whether or not ARM is present. The result of this check determines how CICS initialization proceeds:

+ - If ARM support is not present in MVS, CICS initialization continues normally (that is CICS performs a warm, emergency, or cold start).
+ - If ARM support is present in MVS, CICS registers with ARM automatically.

+ ## Registering with ARM

+ CICS always registers with ARM because CICS needs to know whether it is being restarted by ARM and, if it is, whether or not the restart is with persistent JCL. (The ARM register response to CICS indicates whether or not the same JCL that started the failed region is being used for the ARM restart.) You indicate whether MVS is to use the same JCL or command text that previously started CICS by specifying PERSIST as the *restart_type* operand on the RESTART_METHOD parameter in your automatic restart management policy.

+ When it registers with ARM, CICS passes the value 'SYSCICSb' as the element type, and the string 'SYSCICS_*aaaaaaaa*' as the element name, where *aaaaaaaa* is the CICS applid. Using the applid in the element name means that only one CICS region can successfully register with ARM for a given applid. If two CICS regions try to register with the same applid, the second register is rejected by ARM.

+ ## Waiting for predecessor subsystems

+ During initialization CICS issues an ARM WAITPRED (**wait predecessor**) request to wait, if necessary, for predecessor subsystems (such as DB2 and DBCTL) to become available. This is indicated by message DFHKE0406. One reason for this wait is to ensure that CICS can resynchronize with its partner resource managers for recovery purposes before accepting new work from the network.

+ ## De-registering from ARM

+ During normal shutdown, CICS de-registers from ARM to ensure that it is not automatically restarted. Also, if you want to perform an immediate shutdown and do not want ARM to cause an automatic restart, you can specify the NORESTART option on the PERFORM SHUT IMMEDIATE command.

+ Some error situations that occur during CICS initialization cause CICS to issue a message, with an operator prompt to reply GO or CANCEL. If you reply CANCEL, CICS de-registers from ARM before terminating, because if CICS remained registered, an automatic restart would probably encounter the same error condition.

+ For other error situations, CICS does not de-register, and automatic restarts follow. To control the number of restarts, specify in your ARM policy the number of times ARM is to restart a failed CICS region.

+ ## Failing to register

+ If ARM support is present but the register fails, CICS issues message DFHKE0401. In this case, CICS does not know if it is being restarted by ARM, and therefore it doesn't know whether to override the START parameter to force an emergency restart to preserve data integrity.

+ If CICS fails to register and START=COLD is specified, CICS also issues message DFHKE0408. When CICS is restarting with START=COLD, CICS relies on ARM to determine whether to override the start type and change it to AUTO. Because the REGISTER has failed, CICS cannot determine whether the region is being restarted by ARM, and so does not know whether to override the start type.

Message DFHKE0408 prompts the operator to reply ASIS or AUTO, to indicate the type of start CICS is to perform:

- A reply of ASIS means that CICS is to perform the start specified on the START parameter.
- A reply of AUTO means that CICS is being restarted by ARM, and the type of start is to be resolved by CICS. If the previous run terminated abnormally, CICS will perform an emergency restart.

  Note: A CICS restart can have been initiated by ARM, even though CICS registration with ARM has failed in the restarted CICS.

ARM couple data sets

You must ensure that you define the couple data sets required for ARM, and that they are online and active before you start any CICS region for which you want ARM support. CICS automatic ARM registration fails if the couple data sets are not available at CICS startup.

CICS restart JCL and parameters

Each CICS restart can use the previous startup JCL and system initialization parameters, or can use a new job and parameters.

You cannot specify XRF=YES if you want to use MVS automatic restart management. If the XRF system initialization parameter is changed to XRF=YES for a CICS region being restarted by the ARM, CICS issues message DFHKE0407 to the console then terminates.

CICS START options

You are recommended to specify START=AUTO, which causes a warm start after a normal shutdown, and an emergency restart after failure. (START=AUTO also resolves to a cold start when you start a region for the first time with newly initialized catalogs.)

You are also recommended to always use the same JCL, even if it specifies START=COLD, to ensure that CICS restarts correctly when restarted by the MVS automatic restart manager after a failure. With automatic restart management support,

- If your start-up system initialization parameter specifies START=COLD and
- Your ARM policy specifies that the automatic restart manager is to use the same JCL for a restart following a CICS failure.

CICS overrides the start parameter when restarted by the automatic restart manager and enforces START=AUTO. This is reported by message DFHPA1934, and ensures recoverable data is correctly handled by the resultant emergency restart.

If the ARM policy specifies different JCL for an automatic restart, and that JCL specifies START=COLD, CICS obeys this parameter with a risk of loss of data integrity. Therefore, if you need to specify different JCL to ARM, you should specify START=AUTO to ensure data integrity.

Workload policies

Workloads are started initially by scheduling or automation products.

The components of the workload, and the MVS images capable of running them, are specified as part of the policies for MVS workload manager and ARM. The MVS images must have access to the databases, logs, and program libraries required for the workload.

Administrative policies provide ARM with the necessary information to perform appropriate restart processing. You can define one or more administrative policies, but can have only one active policy for all MVS images in a sysplex. You can modify administrative policies by using an MVS-supplied utility, and can activate a policy with the MVS SETXCF command.

MVS automatic restart management definitions

If you are using CICS on MVS/ESA 5.2, you can exploit the MVS automatic restart management facility that it provides to implement a sysplex-wide integrated automatic restart mechanism.

If you want to use the MVS automatic restart manager facility, you should:

1. Implement automatic restart management on the MVS images that the CICS workload is to run on (see “Implementing MVS automatic restart management.”
2. Ensure that CICS startup JCL used to restart CICS regions is suitable for MVS automatic restart management.

If you have not installed MVS/ESA 5.2, or do not wish to use the MVS automatic restart management facility, you can use XRF to provide restart of failed CICS regions. For information about XRF, see the *CICS/ESA 3.3 XRF Guide*.

Implementing MVS automatic restart management

The task of implementing MVS automatic restart management is part of the overall task of planning for and installing MVS/ESA 5.2. For information about MVS automatic restart management, see *Setting Up a Sysplex*, GC28-1779.

Implementing MVS automatic restart management for CICS generally involves the following steps:

- Ensure that the MVS images available for automatic restarts have access to the databases, logs, and program libraries required for the workload
- Identify those CICS regions for which you want to use automatic restart management
- Define restart processes for the candidate CICS regions
- Define ARM policies for the candidate CICS regions
- Ensure that the system initialization parameter XRF=NO is specified for CICS startup.

Benefits of the MVS automatic restart manager

The main benefits of the MVS automatic restart manager are that it:

- Eliminates the need for operator or automatic package restarts, thereby:
  - Improving restart times
  - Reducing errors
  - Reducing complexity
- Ensures that the workload is restarted on MVS images with spare capacity, by working with MVS workload manager.

Requirements for MVS automatic restart management

To use the automatic restart manager you need the following software:

- MVS/ESA – JES2 Version 5 Release 2 or a later, upward-compatible, release
- MVS/ESA – JES3 Version 5 Release 2 or a later, upward-compatible, release

Changes to CICS externals

There are some changes to CICS externals in support of MVS automatic restart management. These are:

- Changes to the system programming interface (SPI)
- Changes to CICS interfaces with other products
- Changes to CICS-supplied transactions
- Changes to messages and codes
- Changes to trace points.

Changes to the system programming interface (SPI)

A new keyword, NORESTART, is added to the EXEC CICS PERFORM SHUTDOWN command to enable you to inhibit automatic restart.

EXEC CICS PERFORM SHUTDOWN command

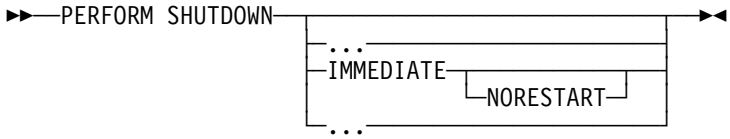
The NORESTART option is provided to allow you to prevent an automatic restart. If you don't want CICS to restart automatically following a SHUTDOWN IMMEDIATE command, specify NORESTART.

NORESTART and TAKEOVER are mutually exclusive, and if issued in the same command, the first of these options is taken and the other ignored. A translator message with severity E (for assembler, C, and PL/I) or C (for COBOL) is issued (return code 8).

Function

The new NORESTART option specifies that CICS is not to be restarted automatically after an immediate shutdown.

Syntax



Conditions:
INVREQ, NOTAUTH

PERFORM SHUTDOWN options

NORESTART

specifies that this CICS region should not be restarted (by MVS automatic restart manager) after the CICS region has completed shutting down.

This option applies to immediate shutdowns only. If you specify NORESTART without IMMEDIATE, CICS performs an immediate shutdown as if you specified the command as PERFORM SHUTDOWN IMMEDIATE NORESTART.

NORESTART (with or without IMMEDIATE) and TAKEOVER are mutually exclusive. If you specify both NORESTART and TAKEOVER, the option named last on the command is ignored and the CICS translator issues an error message (severity E or C, depending on the language used).

Note: CICS can restart automatically only if CICS:

- Is registered with the MVS automatic restart manager (during CICS initialization)
- Has an MVS policy for automatic restart.

TAKEOVER

specifies that this CICS region should be closed down, and that the alternate CICS region is to take over. This option is valid only when the system initialization parameter XRF=YES is specified on CICS startup.

TAKEOVER and NORESTART are mutually exclusive. If you specify both TAKEOVER and NORESTART, the option named last on the command is ignored and the CICS translator issues an error message (severity E or C, depending on the language used).

The other options of the PERFORM SHUTDOWN command remain unchanged from CICS/ESA 3.3.

Changes to CICS interfaces with other products

There are changes to CICS interfaces with DBCTL and VTAM.

Changes for DBCTL

As in earlier releases of CICS, the CICS-supplied module, DFHDXAX, is provided to handle reconnections to DBCTL.

DFHDXAX is one of the modules supplied by CICS for use with DBCTL, known as the IMS control exit routines. The main CICS-supplied IMS control exit is DFHDBCTX, which calls DFHDXAX. For more information about the IMS control exit routines, see the *IMS/ESA Customization Guide: Database*, SC26-3064.

DFHDXAX is provided to ensure that CICS reconnects to DBCTL without operator intervention in the event of a DBCTL restart, or a restart of both CICS and DBCTL. In release earlier than CICS/ESA 4.1, the actions of DFHDXAX are based on the use of a recovery service table (RST), because it was intended for use where either CICS, DBCTL, or both, were running with XRF.

DFHDXAX is extended in CICS/ESA 4.1 to try and avoid operator intervention in those instances when there is not an RST defined; that is, where there is no XRF support. It does this invoking an existing retry loop (based on a timer).

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

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

If an IDENTIFY failed, and CICS is not terminating, the action taken then depends on whether there is an RST defined, which may or may not contain alternative DBCTL ids.

For CICS regions with an RST: If an RST is defined, and it does contain alternative DBCTL ids to which CICS can try to connect, DFHDXAX module selects each DBCTL subsystem ID in the RST in turn as a candidate for reconnection.

The processing, which can take one of two courses, is as described in the *CICS/ESA CICS-IMS Database Control Guide*.

For CICS regions without an RST: If there is not an RST, DFHDXAX selects the current DBCTL ID, and initiates repeated attempts to reconnect to the current DBCTL.

Retries are made every 5 seconds for a ten minute period. If reconnection is still not successful after ten minutes, DFHDXAX abandons the attempt, and requests IMS to issue message DFS0690, which requires operator intervention.

Changes for VTAM

In a VTAM network, the session between CICS and VTAM is started automatically if VTAM is started before CICS. If VTAM is not active when you start CICS, you receive the following messages:

```
+DFHSI1589D 'applid' VTAM is not currently active.
+DFHSI1572 'applid'  Unable to OPEN VTAM ACB - RC=xxxxxxxx, ACB CODE=yy.
```

CICS provides a new transaction, COVR, to open the VTAM ACB automatically when VTAM becomes available. See “The COVR transaction” on page 120 for information about this new CICS-supplied transaction.

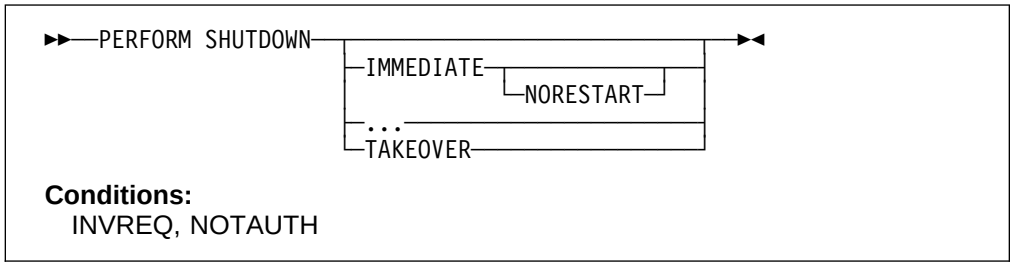
Changes to CICS-supplied transactions

There two changes to CICS-supplied transactions for automatic restart management:

1. The NORESTART option is added to the CEMT PERFORM SHUTDOWN command
2. The COVR transaction is supplied for use with VTAM reconnections.

CEMT PERFORM SHUTDOWN command

Syntax



Conditions:
INVREQ, NOTAUTH

SHUTDOWN options

The NORESTART option on the CEMT PERFORM SHUTDOWN command provides the same function as on the EXEC CICS PERFORM SHUTDOWN command. However, in the case of the CEMT command, the IMMEDIATE option is mandatory with NORESTART. If you specify NORESTART without IMMEDIATE on the CEMT PERFORM SHUT command, CICS ignores the command and issues an error message. (See “Changes to the system programming interface (SPI)” on page 117 for an explanation of NORESTART.)

The COVR transaction

The new transaction COVR, and the program it uses, DFHZCOVR, are introduced in CICS/ESA 4.1.

To ensure that CICS reconnects to VTAM in the event of a VTAM ABEND, CICS keeps retrying the OPEN VTAM ACB using a time-delay mechanism via the non-terminal transaction COVR. After CICS has completed clean-up following the VTAM failure, it invokes the CICS open VTAM retry (COVR) transaction. The COVR transaction invokes the terminal control open VTAM retry program, DFHZCOVR, which performs an OPEN VTAM retry loop with a 5 second wait. CICS issues a DFHZC0200 message every minute, while the open is unsuccessful, and each attempt is recorded on the CSNE transient data queue. After 10 minutes, CICS issues a DFHZC0201 message and terminates the transaction. If CICS shutdown is initiated while the transaction is running, CICS issues a DFHZC0201 message and terminates the transaction.

You cannot run the COVR transaction from a terminal. If you invoke COVR from a termninal, it abends with an AZCU transaction abend.

Problem determination

There are changes to aid problem determination in connection with automatic restart in the form of new messages and additional trace points.

New messages associated with automatic restart

There are some new CICS messages for ARM support, which CICS can issue during startup if problems are encountered when CICS tries to connect to ARM.

The new message numbers are:

DFHKE0401	DFHKE0407
DFHKE0402	DFHKE0408
DFHKE0403	DFHKE0409
DFHKE0404	DFHKE0410
DFHKE0405	DFHZC0200
DFHKE0406	DFHZC0201

For the text of these messages, see the *CICS/ESA Migration Guide*

Changes to trace points

There are some new trace points for DFHKEAR. These trace points are listed in the *CICS/ESA Diagnosis Reference*.

Chapter 8. The IBM CICS Transaction Affinities Utility MVS/ESA

This chapter gives a brief description the IBM CICS Transaction Affinities Utility MVS/ESA, which is provided to help you to identify instances of intertransaction affinity that may hinder your migration to a dynamic transaction routing environment. For an explanation of intertransaction affinities, see the *CICS/ESA Application Programming Guide*.

The IBM CICS Transaction Affinities Utility MVS/ESA is available as a program offering for use with the following releases of CICS:

- CICS/ESA 4.1
- CICS/ESA 3.3
- CICS/ESA 3.2.1
- CICS/MVS 2.1.2

Providing the affinity utility program on the earlier releases enables you to plan how to remove, or how to manage, the intertransaction affinities before migrating to the workload management environment provided by CICS/ESA 4.1.

Prerequisite PTFs

To enable the IBM CICS Transaction Affinities Utility MVS/ESA to work with CICS/ESA 4.1 you need to apply PTF UN61942 to the affinity utility program.

To enable the IBM CICS Transaction Affinities Utility MVS/ESA to work with CICS/MVS 2.1.2 you need to apply PTF UN43826 to CICS/MVS 2.1.2, to provide the CICS global user exit point XEIOUT.

Overview

You can use the IBM CICS Transaction Affinities Utility MVS/ESA to identify the causes of intertransaction affinities in existing CICS/ESA 3.2.1 and CICS/ESA 3.3 workloads on production regions. This helps you to resolve affinities that may prevent effective dynamic transaction routing and workload balancing, and is intended to help you migrate a dynamic transaction routing environment. It can also help you build intertransaction affinities tables for use with CICSPlex System Manager/ESA (CICSPlex SM).

If a dynamic transaction routing program is to function correctly, it must know about any intertransaction affinities if the CICS workload is to be managed effectively. CICS transactions use many different techniques to pass information between one another and to synchronize activity between themselves. Some of these techniques require that the transactions exchanging data or synchronizing activity must execute in the same CICS region, and therefore impose restrictions on the dynamic routing of the transactions. The dynamic transaction routing program must therefore be aware of the transactions that contain affinities so that it can route transactions to the correct application-owning region (AOR).

The affinity utility program consists of the following components:

1. The affinity load module scanner

   This is a batch utility that scans a load module library to detect those programs in the library that issue the EXEC CICS commands that may cause intertransaction affinity.

2. The affinity detector

   This detects intertransaction affinities in an operating CICS region (that is, in real-time) by intercepting the EXEC CICS commands that cause intertransaction affinity, and saving details of the affinities. It consists of two transactions and some global user exit programs.

3. The affinity reporter

   This is a batch utility that takes the affinity data collected by the affinity detector, and generates a report for use by a CICS system programmer. It also generates sets of basic affinity transaction groups for use by the affinity group builder component. An affinity transaction group is a set of transaction identifiers that have an affinity with one another.

4. The affinity group builder

   This is a batch utility that takes the basic affinity transaction groups produced by the reporter, and combines them to produce affinity transaction groups in a form suitable for batch input to CICSPlex SM.

Note: There are some intertransaction affinities that the utility cannot detect (for example, those created by non-CICS functions). For an alternative method of informing your dynamic transaction routing program about intertransaction affinities, see “Receiving information from a routed transaction” on page 103.

For full details of the IBM CICS Transaction Affinities Utility MVS/ESA see the *IBM CICS Transaction Affinities Utility MVS/ESA: User's Guide*.

Part 3. Solution enablement features

This Part describes solution enablement features in CICS/ESA 4.1, under the following topics:

Chapter 9. Non-terminal security

This chapter describes new mechanisms provided for non-terminal security in CICS/ESA 4.1. It covers the following topics:

- Overview
- Benefits of non-terminal security
- Changes to CICS externals.

The descriptions of non-terminal security in this chapter are based on the function provided by RACF.

Overview

In CICS/ESA 4.1, security is extended in a number of ways to transactions that do not have a terminal associated with them. This is referred to as non-terminal security.

The non-terminal security function described in this chapter exploits changes made in the security and user domains. For details of these, see Chapter 22, “Security manager domain” on page 385 and Chapter 23, “User domain” on page 401.

Security for transactions started by the START command

To ensure that there is always a userid associated with a started transaction, even when the START command does not specify a terminal, CICS propagates the userid from the initiating transaction to the started transaction. This enables CICS to perform a transaction-attach security check, and resource and command security checks, on the started task.

You can override the propagated userid by specifying a userid explicitly on the EXEC CICS START command.

Security for transient data trigger-level transactions

In CICS/ESA 4.1, you can specify a userid explicitly for those transactions that are started, without a terminal, when a transient data queue reaches its trigger level.

By default, trigger-level transactions run under the CICS default userid specified on the DFLTUSER system initialization parameter.

Security for program list table (PLT) programs at startup

You can now specify a userid for programs that are defined in a post-initialization program list table (PLTPI). These run under a new CICS-supplied transaction, CPLT, for which the userid is specified in new system initialization parameters.

As in earlier releases, CICS/ESA 4.1 continues to support security for shutdown PLT programs, whereby PLT programs run during shutdown under the authority of the userid that initiates the shutdown.

Surrogate user checking

Non-terminal security and PLTPI security are subject to surrogate user checking, a facility provided by the security domain. (see Chapter 22, “Security manager domain” on page 385).

CICS performs surrogate user security checking in a number of instances, to ensure that the surrogate user is authorized to act for the other user.

For more information about defining surrogate user profiles, see the *Resource Access Control Facility (RACF) Security Administrator's Guide*.

Benefits of non-terminal security

Prior to CICS/ESA 4.1, the userid for a transaction could be determined only for those transactions associated with a terminal. CICS used this terminal-related userid for security checking at transaction-attach, and for command and resource security.

This dependency on the association with a terminal meant that resources used in transactions not associated with a terminal could not be protected against unauthorized use. Transaction attach always succeeded for a started transaction without a terminal, but any resource and or command security checks always failed.

In CICS/ESA 4.1, the security mechanisms are extended to protect such transactions that were previously exposed. These extensions, which are designed to satisfy the most stringent security audit requirements, provide security for

- Started non-terminal transactions
- Transient data trigger-level transactions
- PLT programs that run during CICS initialization.

Changes to CICS externals

The provision of non-terminal security in CICS/ESA 4.1 results in a number of changes to CICS externals:

- Changes to system definition
- Changes to the application programming interface
- Changes to the system programming interface
- Changes to CICS-supplied transactions
- Changes to resource definition macros
- Changes to clobal user exits

This section describes the changes to these CICS externals that are needed to protect resources used in non-terminal transactions and in PLT programs that run during CICS initialization.

Changes to system definition

The PLTPIUSR and PLTPISEC system initialization parameters are introduced to specify security options for PLT programs that are run during CICS initialization.

Table 7. The DFHSIT macro parameters		
	DFHSIT	[TYPE={**CSECT**|DSECT}] ⋮ [,PLTPISEC={**NONE**|CMDSEC|RESSEC|ALL}] [,PLTPIUSR=userid] ⋮
	END	DFHSITBA

PLTPISEC={NONE|CMDSEC|RESSEC|ALL}

Specifies whether or not you want CICS to perform command security or resource security checking for PLT programs during CICS initialization. The PLT programs run under the authority of the userid specified on PLTPIUSR, which must be authorized to the appropriate resources defined by PLTPISEC.

NONE

specifies that you do not want any security checking on PLT initialization programs.

CMDSEC

specifies that you want CICS to perform command security checking only.

RESSEC

specifies that you want CICS to perform resource security checking only.

ALL

specifies that you want CICS to perform both command and resource security checking.

Restrictions

You can code the PLTPISEC parameter in the SIT, PARM, or SYSIN only.

PLTPIUSR=userid

Specifies the userid that CICS is to use for security checking for PLT programs that run during CICS initialization. All PLT programs run under the authority of the specified userid, which must be authorized to all the resources referenced by the programs, as defined by the PLTPISEC parameter.

PLT programs are run under the CICS internal transaction, CPLT. Before the CPLT transaction is attached, CICS performs a surrogate user check against the CICS region userid (the userid under which the CICS region is executing). This is to ensure that the CICS region is authorized as a surrogate of the userid specified on the PLTPIUSR parameter. This ensures that you cannot arbitrarily specify any PLT userid in any CICS region—each PLT userid must first be authorized to the appropriate CICS region.

If you do not specify the PLTPIUSR parameter, CICS runs PLTPI programs under the authority of the CICS region userid, in which case CICS does not perform a surrogate user check. However, the CICS region userid must then be authorized to all the resources referenced by the PLT programs.

Restrictions

You can code the PLTPIUSR parameter in the SIT, PARM, or SYSIN only.

Changes to the application programming interface

The EXEC CICS START command is extended to handle security for non-terminal transactions started by a START command. All transactions started by the START command have an associated security environment, which is either inherited from the transaction issuing the START command, or is specified by the USERID option on the command.

The following changes are made to the START command:

- New USERID option
- New USERIDERR condition
- Purposes of NOTAUTH condition is extended.

To verify that the userid of one transaction is authorized to issue a START command for another transaction that specifies a different userid, CICS uses RACF surrogate user checking.

CICS can issue up to three surrogate user security checks on a single START command, depending on the circumstances. These can be against the following userids:

1. The userid of the transaction that issues the START command, if USERID is specified
2. The userid of the CEDF transaction, if the transaction that issues the START command is being run in CEDF dual-screen mode
3. The security userid in effect for a communication link, if:
   - The START command is executed in a remote system by means of function shipping or transaction routing, and
   - Link security is in effect.

   The security userid in the remote case can be derived from one of a number of sources.
   - For ISC links, it is either:
     - The USERID on the SESSIONS definition, or
     - The SECURITYNAME on the CONNECTION definition.
   - For MRO links, it is either:
     - The USERID on the SESSIONS definition, or
     - The CICS region userid of the MRO partner.

A separate surrogate user security check is done for each of these userids, as required, before the transaction is attached.

+ Changes affecting the EXEC CICS ASSIGN command

+ In earlier releases, the OPID, OPCLASS, NATLANGINUSE, USERID, USERNAME, and USERPRIORITY options of the ASSIGN command returned the INVREQ condition if the command was issued in a transaction that was not associated with a terminal. The OPID and USERID options of the EXEC CICS ASSIGN command no longer return the INVREQ condition, and always return a value. The value returned originates from a number of sources depending on the option.

+ The remaining four options (OPCLASS, NATLANGINUSE, USERNAME, and
+ USERPRIORITY) no longer return the INVREQ condition except when the
+ command is issued in a program invoked by a distributed program link command.

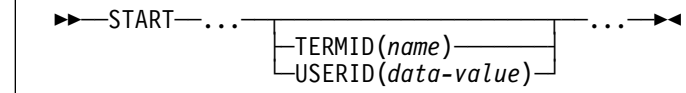
EXEC CICS START command

General-use programming interface

Function

The START command is enhanced to allow you to specify a userid for security checking in the started transaction if you do not specify an associated terminal.

Syntax



Conditions:
..., INVREQ, NOTAUTH, ..., USERIDERR

START options

USERID(data-value)

Specifies the userid under whose authority the started transaction is to run, if the started transaction is not associated with a terminal (that is, when TERMID is not specified). This is referred to as *userid1*.

If you omit both TERMID and USERID, CICS uses instead the userid under which the transaction that issues the START command is running. This is referred to as *userid2*.

By using either *userid1* or *userid2* CICS ensures that a started transaction always runs under a valid userid, which must be authorized to all the resources referenced by the started transaction.

CICS performs a surrogate security check against *userid2* to verify that this user is authorized to *userid1*. If *userid2* is not authorized, CICS returns a NOTAUTH condition. The surrogate check is not done here if USERID is omitted.

START conditions

INVREQ

the INVREQ condition occurs if USERID is specified and the CICS external security manager interface is not initialized (RESP2=18)

Default action: Terminate the task abnormally.

NOTAUTH

the NOTAUTH condition occurs if a surrogate user check fails; that is, the user (*userid2*) is not defined to the external security manager as a surrogate for the user (*userid1*) specified on the START command (RESP2=9).

The scope of the surrogate user check(s) depend on whether the task is running locally, under the CEDF transaction, or is running under link security.

Default action: Terminate the task abnormally.

USERIDERR
occurs in any of the following situations:

- The specified USERID is not known to the external security manager (RESP2=8).
- The external security manager is in a state such that CICS cannot determine whether a specified USERID is valid (RESP2=10).

Default action: Terminate the task abnormally.

Changes to the system programming interface

A new option, ATIUSERID, is added to the INQUIRE and SET TDQUEUE commands.

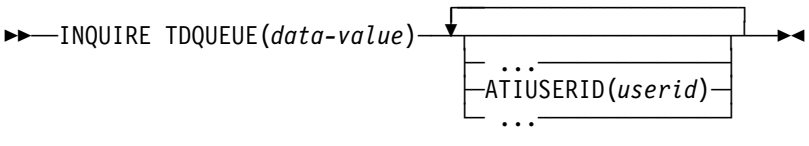
The reasons for the INVREQ and NOTAUTH conditions are extended for the SET command.

EXEC CICS INQUIRE TDQUEUE command

Function

Allows you to obtain the userid on a transient data trigger-level transaction.

Syntax



Conditions:
NOTAUTH

INQUIRE TDQUEUE options

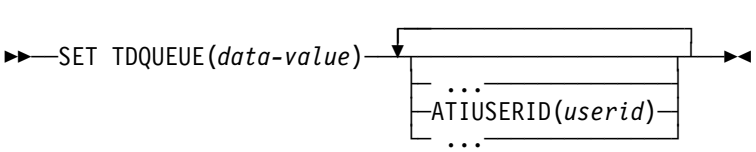
ATIUSERID(data-area) (intrapartition queues only)
returns, as an 8-byte string, the userid for a transaction that is started without an associated terminal when the trigger level is reached.

EXEC CICS SET TDQUEUE command

Function

Allows you to specify a userid on a transient data trigger-level transaction.

Syntax



Conditions:
INVREQ, NOTAUTH, USERIDERR

SET TDQUEUE options

ATIUSERID(data-value) (intrapartition queues only)

specifies the userid for a transient data trigger-level transaction that is not associated with a terminal. The name can be up to 8 bytes long.

The ATIFACILITY must be NOTERMINAL.

CICS performs a surrogate user security check against the userid of the transaction that issues the SET TDQUEUE command, to verify that the transaction userid is authorized to the userid specified on the ATIUSERID parameter.

SET TDQUEUE conditions

INVREQ

this condition occurs if:

- ATIFACILITY is specified as TERMINAL and an ATIUSERID is specified (RESP2=17).
- ATIUSERID is specified for an intrapartition queue that is not defined with DESTFAC=FILE or the ATIFACILITY of the queue has been changed to TERMINAL (RESP2=18).
- ATIUSERID is specified for an extrapartition queue (RESP2=19).
- The CICS external security manager interface is not initialized (RESP2=20).
- CICS has received an unknown response from the ESM (RESP2=21).
- The external security manager is not responding (RESP2=22).

Default action: Terminate the task abnormally.

NOTAUTH

this condition occurs if:

- The userid is not authorized to the RACF APPL profile for this CICS region. See the *CICS/ESA CICS-RACF Security Guide* for information about authorizing users to the RACF APPL class profile. (RESP2=23).
- The specified userid is revoked (RESP2=24).
- The userid specified is found to be not authorized during SECLABEL processing by the external security manager. For information about security labels, see the *Resource Access Control Facility (RACF) Security Administrator's Guide*. (RESP2=25).
- The userid's access to the current group is revoked (RESP2=27).
- The user of the transaction issuing the SET TDQUEUE command is not an authorized surrogate of the userid on ATIUSERID (RESP2=102).

Default action: Terminate the task abnormally.

USERIDERR

this condition occurs if the specified userid is not known to the external security manager (RESP2=28).

End of General-use programming interface

Changes to CICS-supplied transactions

There is a new CICS-supplied transaction, CPLT, which is attached during CICS startup to run PLT programs defined in a PLTPI table.

New security mechanisms, and the PLTPISEC and PLTPIUSR system initialization parameters, are added to allow you to specify security checks for the CPLT transaction.

Changes to resource definition (macro)

To enable you to specify security for transactions started by transient data trigger-levels, a new USERID parameter is added to the DFHDCT macro.

Changes to DFHDCT macro

You can specify the USERID option, described below, on DFHDCT TYPE=INITIAL or TYPE=INTRA macros. CICS uses the userid specified on DCT macros for security checking in any trigger-level transactions that are not associated with a terminal.

On the TYPE=INITIAL macro

USERID=userid

Code this with the userid that you want CICS to use for security checking for any trigger-level TYPE=INTRA entry that does not specify its own userid.

If you omit the userid from a trigger-level entry, and also from this TYPE=INITIAL macro, CICS uses the CICS default userid, specified on the DFLTUSER system initialization parameter.

See the USERID description for the TYPE=INTRA macro for information about surrogate user checking.

On the TYPE=INTRA macro

USERID=userid

Code this with the userid that you want CICS to use for security checking for the trigger-level transaction specified on the TRANSID operand. USERID is valid only when the destination is defined as DESTFAC=FILE.

The trigger-level transaction runs under the authority of the specified userid, which must be authorized to all the resources used by the transaction.

If you omit the userid from a qualifying trigger-level entry, CICS uses the userid specified on the TYPE=INITIAL macro. If you omit the userid from the TYPE=INITIAL macro also, CICS uses the CICS default userid, specified on the DFLTUSER system initialization parameter. You must ensure that the CICS region userid, of any CICS region in which this DCT is installed, is defined as a surrogate of all the userids specified in the DCT. This is because, during initialization, CICS performs a surrogate user security check against the CICS region userid. If the surrogate security check fails, CICS deactivates automatic transaction initiation by trigger-level for the intrapartition queue for which the surrogate check failed.

For more information about surrogate user security, see the *CICS/ESA CICS-RACF Security Guide*.

Changes to global user exits

The userid is passed as an additional input parameter to the XICEREQ and XICEXP global user exits.

Chapter 10. New and enhanced EXEC interface global user exits

This chapter describes the new global user exits, introduced in CICS/ESA 4.1, for the EXEC interface layer in the application programming (AP) domain. It also describes the enhancements made to the existing EXEC interface global user exits. It covers the following topics:

- Overview
- Benefits of the new and enhanced exits
- Changes to CICS externals.

This chapter contains Product-sensitive Programming Interface information.

Overview

There are new EXEC interface global user exits added to the AP domain for the CICS interval control, temporary storage, and transient data facilities. These exits are as follows:

Interval control	Exits XICEREQ and XICEREQC
Transient data	Exits XTDEREQ and XTDEREQC
Temporary storage	Exits XTSEREQ and XTSEREQC

These new global user exits follow the same design as the EXEC interface global user exits added in earlier releases for file control (XFCREQ and XFCREQC) and program control (XPCREQ and XPCREQC). Like the earlier exits, they are designed to give control to your global user exit program immediately before, and immediately after, the execution of the relevant CICS API command.

Changes to existing EXEC interface global user exits

The change introduced in CICS/ESA 4.1 to enable the new EXEC interface modules to use most of the API and all of the SPI commands is extended to the EXEC interface modules of earlier releases. This enhancement applies to the following exit points:

- XDBDERR and XDBFERR
  XDBIN and XDBINIT — Dynamic transaction backout
- XFCREQ and XFCREQC — File control
- XICEREQ and XICEREQC — Interval control
- XPCREQ and XPCREQC — Program control
- XTDEREQ and XTDEREQC — Transient data
- XTSEREQ and XTSEREQC — Temporary storage
- XRCFCER and XRCINIT
  XRCINPT and XRCOPER — Restart transaction backout
- XRMIIN and XRMIOUT — Resource manager interface

To enable you to control the use of the API and guard against recursion, a new parameter, UEPRECUR, is added to all the EXEC interface exits. UEPRECUR is the address of a recursion count, which you are recommended to use to avoid loops.

New interval control global user exits

The interval control global user exits satisfies a user requirement to be able to modify the following CICS API interval control commands before they are processed:

- CANCEL
- DELAY
- POST
- RETRIEVE
- START
- ASKTIME

CICS/ESA 4.1 provides two new global user exit points for interval control requests, XICEREQ and XICEREQC:

XICEREQ Global user exit programs enabled for this exit point are invoked before CICS processes one of the applicable interval control API requests.

XICEREQC Global user exit programs enabled for this exit point are invoked on completion of each applicable interval control API request.

New transient data global user exits

The transient data global user exits are invoked for all the CICS API transient data commands. These are:

- DELETEQ TD
- READQ TD
- WRITEQ TD

CICS/ESA 4.1 provides two new global user exit points for transient data requests:

XTDEREQ Global user exit programs enabled for this exit point are invoked before CICS processes a transient data API request.

XTDEREQC Global user exit programs enabled for this exit point are invoked on completion of each transient data API request.

Temporary storage global user exits

Full CICS/ESA 4.1 exploitation of workload management to improve overall performance requires dynamic routing of transactions in order to balance work between many CICS application-owning regions (AORs) running in multiple MVS images. Some application programs use temporary storage as a means of passing data between transactions. This means that such transactions must have access to the same temporary storage queue. If temporary storage queues are local to an AOR, and if such local temporary storage queues are widely used, they inhibit the scope of the CICS dynamic transaction routing capability. Transactions that share local temporary storage force a dynamic transaction routing program to route the transactions to the same AOR.

CICS/ESA 4.1 provides two new global user exit points for temporary storage requests:

XTSEREQ Global user exit programs enabled for this exit point are invoked before CICS processes a temporary storage API request.

XTSEREQC Global user exit programs enabled for this exit point are invoked on completion of each temporary storage API request.

The XTSEREQ global user exit programs can modify API requests so that the commands can be executed locally or function shipped. For example, the user exit program can add the SYSID option to a previously local TS request, causing CICS to route the request to a remote CICS region. This enables CICS to exploit dynamic transaction routing and workload balancing while, at the same time, allowing transactions to continue to use temporary storage queues as a means of passing data. It also enables you to create separate temporary storage queue-owning regions (QORs) for global use within a CICSplex.

Benefits of the new AP domain global user exits

The changes to the EXEC interface global user exits allow system programmers to use all the EXEC CICS API commands (except EXEC CICS RETURN and EXEC CICS ABEND), and all the EXEC CICS SPI commands. This applies to all the EXEC interface global user exits of earlier releases as well as the new EXEC interface exits introduced in CICS/ESA 4.1.

Allowing EXEC interface exit programs to use virtually the full scope of the command-level programming interface, however, increases the risk of recursion in these exits. To help you control the use of the API and SPI commands, and to avoid recursion, these exits all include the UEPRECUR parameter, which passes a recursion count to your exit programs.

Avoiding temporary storage inter-transaction affinities

You can use the new temporary storage global user exit programs to enable existing transactions, which may have inter-transaction affinities relating to their use of temporary storage queues, to run unmodified in a dynamic transaction routing environment without modification.

Unlike the existing temporary storage queues, the new temporary storage global user exit, XTSEREQ, is invoked *before* CICS checks whether a queue referenced by your application is local or remote. This means that an XTSEREQ global user exit program can intervene and, if necessary, modify temporary storage queue names. This enables you to change queue names so that they conform to a naming convention that allows them to be defined generically in a temporary storage table, as described in the *CICS/ESA Resource Definition Guide*.

Changes to CICS externals

There are no changes to CICS externals other than the changes to the global user exit interface by the introduction of the new points, and the addition of a new parameter, UEPSRCTK, for the XMNOUT user exit:

UEPSRCTK Address of the MVS workload manager service reporting class token for the current transaction. If CICS support for MVS workload management is not available, this token is null.

Interval control program exits XICEREQ and XICEREQC

XICEREQ is invoked on entry to the interval control program before CICS processes an interval control request. Using XICEREQ, you can:

- Analyze the request to determine its type, the keywords specified, and their values.
- Modify any value specified by the request before the command is executed.
- Set return codes to specify that either:
  - CICS should continue with the request, modified or unmodified.
  - CICS should bypass the request. (Note that if you set this return code, you must also set up return codes for the EXEC interface block (EIB), as if you had processed the request yourself.)

XICEREQC is invoked after the interval control program request is completed. Using XICEREQC, you can:

- Analyze the request, to determine its type, the keywords specified, and their values.
- Set return codes for the EIB.

CICS passes six parameters to these exits as follows:

- The address of the command-level parameter structure (UEPCLPS)
- The address of a token (UEPICTOK) used to pass 4 bytes of data from XICEREQ to XICEREQC
- The addresses of copies of three pieces of return code information from the EIB
- The address of a token (UEPTSTOK) that is valid throughout the life of a task
- The address of an exit recursion count (UEPRECUR).

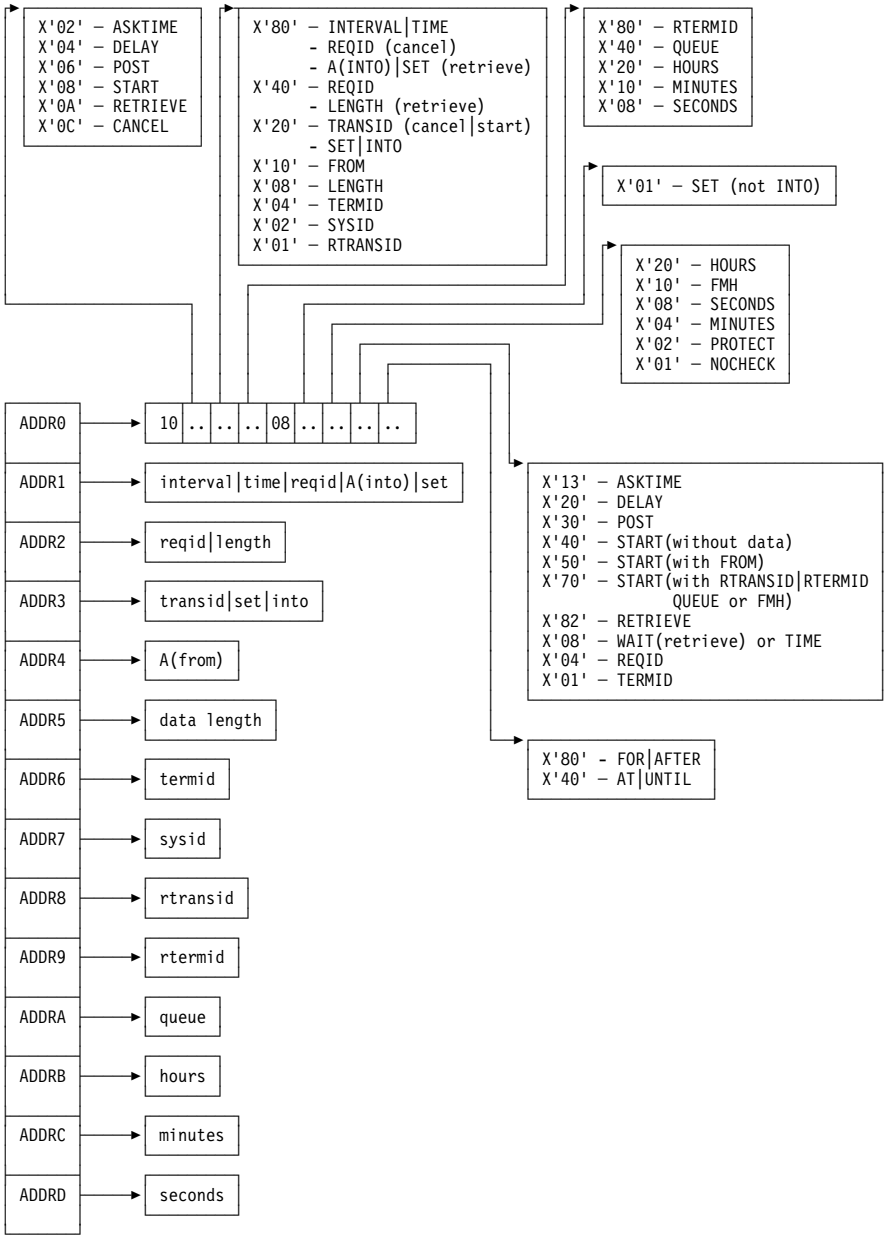


Figure 10. The command-level parameter structure for interval control

The command-level parameter structure consists of a series of addresses. The first address points to the EXEC interface descriptor (EID), which consists of a 9-byte area that describes the type of request and identifies each keyword specified with the request. The remaining addresses point to pieces of data associated with the request. For example, the second address points to the interval for START requests.

You can examine the EID to determine the type of request and the keywords specified. You can examine the other parameters in the list to determine the values of the keywords. You can also modify values of keywords specified on the request. For example, you could change the SYSID specified in the request.

End of parameter list indicator

The high-order bit is set on in the last address set in the parameter list to indicate that it is the last one in the list. On return from your user exit program, CICS scans the parameter list for the high-order bit to find the last parameter. Therefore, if you modify the length of the parameter list, you must also reset the high-order bit to indicate which is the new last address.

For example, if the parameter list specifies only the first four addresses (IC_ADDR0, the address of the EID, to IC_ADDR3, the address of the name of the transaction named in a START request), the high-order bit is set on in IPC_ADDR3. If you extend the parameter list by setting the address of a SYSID in IC_ADDR7, you must unset the high-order bit in IC_ADDR3 and set it on in IC_ADDR7 instead.

The maximum size of parameter list is supplied to the exit, thus allowing your exit program to add any parameters not already specified without needing to first obtain more storage.

The original parameter list, as it was before XICEREQ was invoked, is restored after the completion of XICEREQC. It follows that the execution diagnostic facility (EDF) displays the original command before **and** after execution: **EDF does not display any changes made by the exit**.

The UEPCLPS exit-specific parameter: The UEPCLPS exit-specific parameter is included in both exit XICEREQ and exit XICEREQC. It is the address of the command-level parameter structure. The command-level parameter structure contains 14 addresses, IC_ADDR0 through IC_ADDRD. These are described in the DSECT DFHICUED, which you should copy into your exit program by including the statement COPY DFHICUED.

The command-level parameter list is made up as follows:

IC_ADDR0
is the address of a 9-byte area called the EXEC interface descriptor (EID), which is made up as follows:

IC_GROUP
IC_FUNCT
IC_BITS1
IC_BITS2
IC_EIDOPT5
IC_EIDOPT6
IC_EIDOPT7
IC_EIDOPT8
IC_EIDOPT9
IC_EIDOPT10
IC_EIDOPT11
IC_EIDOPT12
IC_EIDOPT13

IC_GROUP Always X'10', indicating that this is an interval control request.

IC_FUNCT One byte that defines the type of request:

X'02' ASKTIME
X'04' DELAY
X'06' POST
X'08' START
X'0A' RETRIEVE
X'0C' CANCEL

IC_BITS1 Existence bits that define which arguments were specified. To obtain the argument associated with a keyword, you need to use the appropriate address from the command-level parameter structure. Before using this address, you must check the associated existence bit. If the existence bit is set off, the argument was not specified in the request and the address should not be used.

X'80' Set if the request contains INTERVAL or TIME arguments, or if a CANCEL request specifies REQID, or if a RETRIEVE request specifies SET or INTO. If set, **IC_ADDR1** is meaningful.
X'40' Set if the request other than CANCEL specifies REQID or if a RETRIEVE request specifies LENGTH. If set, **IC_ADDR2** is meaningful.
X'20' Set if the request specifies TRANSID or if a request other than RETRIEVE specifies SET or INTO. If set, **IC_ADDR3** is meaningful.
X'10' Set if the request specifies FROM. If set, **IC_ADDR4** is meaningful.
X'08' Set if a request other than RETRIEVE specifies LENGTH. If set, **IC_ADDR5** is meaningful.
X'04' Set if the request specifies TERMID. If set, **IC_ADDR6** is meaningful.
X'02' Set if the request specifies SYSID. If set, **IC_ADDR7** is meaningful.
X'01' Set if the request specifies RTRANSID. If set, **IC_ADDR8** is meaningful.

IC_BITS2 Further argument existence bits.

X'80' Set if the request specifies RTERMID. If set, **IC_ADDR9** is meaningful.
X'40' Set if the request specifies QUEUE. If set, **IC_ADDRA** is meaningful.
X'20' Set if the request specifies HOURS. If set, **IC_ADDRB** is meaningful.
X'10' Set if the request specifies MINUTES. If set, **IC_ADDRC** is meaningful.
X'08' Set if the request specifies SECONDS. If set, **IC_ADDRD** is meaningful.

IC_BITS3 One byte not used by interval control.

IC_EIDOPT5 Indicates whether certain keywords were specified on the request.

X'01' SET (and not INTO) was specified. You cannot modify this field in your user exit.

IC_EIDOPT6 Indicates whether certain keywords were specified on the request.

X'02'	HOURS specified.
X'04'	FMH specified.
X'06'	SECONDS specified.
X'08'	MINUTES specified.
X'0A'	PROTECT specified.
X'0C'	NOCHECK specified.

IC_EIDOPT7 Indicates whether certain functions or keywords were specified on the request.

X'F0'	CANCEL specified.
X'82'	RETRIEVE specified.
X'40'	START specified.
X'30'	POST specified.
X'20'	DELAY, RTRANSID, RTERMID, or QUEUE specified, and/or FMH.
X'13'	ASKTIME specified.
X'10'	FROM, RTRANSID, or RTERMID specified, and/or QUEUE.
X'08'	TIME or WAIT specified.
X'04'	REQID specified.
X'01'	TERMID specified.

IC_EIDOPT8 Indicates whether certain keywords were specified on the request.

X'80'	FOR or AFTER specified.
X'40'	AT or UNTIL specified.

IC_ADDR1
is the address of one of the following:

- An 8-byte area containing the value of the INTERVAL keyword (or TIME keyword if **IC_EIDOPT7** indicates that TIME is specified).
- An 8-byte area containing the value of REQID (if the request is CANCEL).
- Data from INTO (if the request is RETRIEVE, and if **IC_EIDOPT5** indicates that this is not SET).
- A 4-byte address from SET (if the request is RETRIEVE and **IC_EIDOPT5** indicates that this is SET).

IC_ADDR2
is the address of one of the following:

- An 8-byte area containing the value of REQID (if the request is DELAY, POST or START).
- A halfword containing the value of LENGTH (if the request is RETRIEVE).

  Warning: For requests that specify INTO, do not change the value of LENGTH to a value greater than that specified by the application. To do so causes a storage overlay in the application.

IC_ADDR3
is the address of one of the following:

- An area containing the value of TRANSID (if the request is CANCEL or START).
- Data from INTO (if the request is START and **IC_EIDOPT5** indicates that this is not SET).
- A 4-byte address from SET (if the request is START or POST and **IC_EIDOPT5** indicates that this is SET).

IC_ADDR4
is the address of an area containing the data from FROM.

IC_ADDR5
is the address of the halfword value of LENGTH.

Warning: For requests that specify INTO, do not change the value of LENGTH to a value greater than that specified by the application. To do so causes a storage overlay in the application.

IC_ADDR6
is the address of an area containing the value of TERMID.

IC_ADDR7
is the address of an area containing the value of SYSID.

IC_ADDR8
is the address of an area containing the value of RTRANSID.

IC_ADDR9
is the address of an area containing the value of RTERMID.

IC_ADDRA
is the address of an area containing the value of QUEUE.

IC_ADDRB
is the address of an area containing the value of HOURS.

IC_ADDRC
is the address of an area containing the value of MINUTES.

IC_ADDRD
is the address of an area containing the value of SECONDS.

Modifying fields in the command-level parameter structure: Some fields that are passed to interval control are used as input to the request, some are used as output fields, and some are used for both input and output. The method your user exit program uses to modify a field depends on the usage of the field.

The following are always input fields:

INTERVAL
TIME
REQID
FROM
LENGTH
TERMID
SYSID

HOURS
MINUTES
SECONDS

The following are always output fields:

INTO
SET

The following are input fields on the START request and output fields on the RETRIEVE request:

RTRANSID
RTERMID
QUEUE

Modifying input fields: The correct method of modifying an input field is to create a new copy of it, and to change the address in the command-level parameter list to point to your new data. **Warning:** You must never modify an input field by altering the data that is pointed to by the command-level parameter list. To do so would corrupt storage belonging to the application program and would cause a failure when the program attempted to reuse the field.

Modifying output fields: The technique described in "Modifying input fields" is not suitable for modifying output fields. (The results would be returned to the new area instead of the application's area, and would be invisible to the application.)

An output field is modified by altering the data that is pointed to by the command-level parameter list. In the case of an output field, you can modify the application's data in place, because the application is expecting the field to be modified anyway.

Modifying the EID: It is not possible to modify the EID to make major changes to requests, such as changing a DELAY request to a START request.

However, you can make minor changes to requests, such as turning on the existence bit for SYSID so that the request can be changed into one that is shipped to a remote system.

Some interval control commands use 2 bits in the EID to indicate a single keyword; the EXEC CICS START command, for example, uses 2 bits to indicate TERMID. The first bit, in IC_BITS1, indicates that ADDR6 in the command parameter list is valid (ADDR6 points to TERMID) and the second, in IC_EIDOPT7, is the keyword existence bit to show that the TERMID keyword was specified on the command.

Where this occurs you must ensure that both bit settings are changed (consistently) if you wish to modify these commands from within a user exit program, or the results will be unpredictable.

The list that follows shows the bits in the EID that **can** be modified. Any attempt to modify any other part of the EID is ignored.

IC_BITS1

X'80' The existence bit for REQID (if the request is CANCEL)

X'40' The existence bit for LENGTH (if the request is RETRIEVE) or REQID
X'10' The existence bit for FROM
X'08' The existence bit for LENGTH
X'04' The existence bit for TERMID
X'02' The existence bit for SYSID
X'01' The existence bit for RTRANSID

IC_BITS2

X'80' The existence bit for RTERMID
X'40' The existence bit for QUEUE
X'20' The existence bit for HOURS
X'10' The existence bit for MINUTES
X'08' The existence bit for SECONDS

IC_EIDOPT6

X'20' Secondary existence bit for HOURS
X'10' Existence bit for FMH
X'08' Secondary existence bit for SECONDS
X'04' Secondary existence bit for MINUTES
X'02' Existence bit for PROTECT
X'01' Existence bit for NOCHECK

IC_EIDOPT7

Bits in IC_EIDOPT7 should only be modified within the same functional group – that is, only those existence bits defined as valid for a START request should be set on a START request.

ASKTIME requests

X'13' ASKTIME request. This value is fixed for all ASKTIME requests, and should not be modified.

DELAY requests

X'20' DELAY request
X'08' TIME specified
X'04' REQID specified

POST requests

X'30' POST request
X'08' TIME specified
X'04' REQID specified

START requests

X'40' START request (without DATA)
X'50' START with DATA request
X'70' START with one or more of RTRANSID, RTERMID, QUEUE, or FMH specified.
X'08' TIME specified
X'04' REQID specified
X'01' TERMID specified

RETRIEVE requests

X'82' RETRIEVE request
X'08' TIME specified

CANCEL requests

X'F0' CANCEL request
X'04' REQID specified

Your exit program may provide its own command-level parameter structure and its own EID. If it provides its own command-level parameter structure, you should modify UEPCLPS to point to the new structures. If it provides its own EID, you should modify IC_ADDR0 to point to the new structures.

The EID is reset to its original value before return to the application program. That is, changes made to the EID are retained for the duration of the interval control request only. **Warning:** Your user exit program is prevented from making major changes to the EID. However, you must take great care when making the minor modifications that **are** permitted.

Using the interval control request token UEPICTOK: UEPICTOK provides the address of a 4-byte area that you can use to pass information between the XICEREQ and XICEREQC user exits for the same interval control request. For example, the address of a piece of storage obtained by the XICEREQ user exit, which is to be freed by the XICEREQC exit, can be passed in the UEPICTOK field.

The EIB: Copies of EIBRCODE, EIBRESP, and EIBRESP2 are passed to the exit, so that you can:

- Modify or set completion information in XICEREQ and XICEREQC
- Examine completion information in XICEREQC.

You can update the copies of EIBRCODE, EIBRESP, and EIBRESP2 that you are given in the parameter list. Interval control copies your values into the real EIB after the completion of XICEREQC; or if you specify a return code of ‘bypass’ in XICEREQ.

You must set valid interval control responses. You must set all three of EIBRCODE, EIBRESP, and EIBRESP2 to a consistent set of values, such as would be set by CICS interval control to describe a valid completion. **CICS does not police the consistency of EIBRCODE, EIBRESP, and EIBRESP2**. However, if EIBRCODE is set to a non–zero value and EIBRESP is set to zero, CICS overrides EIBRESP with a non–zero value. To aid you in setting the values of EIBRCODE, EIBRESP, and EIBRESP2, the values used by interval control are specified in DFHICUED.

Example of how XICEREQ and XICEREQC can be used: XICEREQ and XICEREQC can be used for a variety of purposes. One example of a possible use is given below.

In this example, XICEREQ and XICEREQC are used to route START requests to a number of different CICS regions to provide a simple load balancing mechanism. The example shows only the capabilities of the exits; it is not intended to indicate an ideal way of achieving the function.

In XICEREQ:

1. Scan the global work area (GWA) to locate a suitable CICS region (for example, the region currently processing the least number of START requests).
2. Having decided which system to route the request to, increment the use count for this system.
3. Obtain a 4-byte area in which to store the SYSID for this request. This can be allocated from the GWA to avoid issuing a GETMAIN. If the area is obtained by issuing a GETMAIN, set UEPICTOK to the address of the storage obtained.
4. Set IC_ADDR7 to be the address of the 4–byte area so that XICEREQC can also use this area.
5. If setting IC_ADDR7 now makes it the last address, set the high-order bit in the address, and reset the high-order bit in what was previously the last address.
6. Set the `X'02'` existence bit on in IC_BITS1 to indicate that a SYSID is specified.
7. Return to CICS.

In XICEREQC:

1. Scan the global work area (GWA) and locate the entry for the CICS region specified in the SYSID parameter.
2. Decrement the use count for this system.
3. If a GETMAIN was issued in XICEREQ to obtain an area to hold the SYSID, issue a FREEMAIN for the address held in UEPICTOK.
4. Return to CICS.

Exit XICEREQ

	Invoked before CICS processes an interval control API request.	
Exit-specific parameters	**UEPCLPS**	Address of the command-level parameter structure. See “The UEPCLPS exit-specific parameter” on page 142.
	UEPICTOK	Address of the 4-byte token to be passed to XICEREQC. This allows you, for example, to pass a work area to exit XICEREQC.
	UEPRCODE	Address of a 6-byte hexadecimal copy of the EIB return code ‘EIBRCODE’. For details of EIB return codes, refer to the *CICS/ESA Application Programming Reference* manual.
	UEPRESP	Address of a 4-byte binary copy of the EIB response code ‘EIBRESP’.
	UEPRESP2	Address of a 4-byte binary copy of the EIB response code ‘EIBRESP2’.
	UEPTSTOK	Address of a 4-byte token that is valid throughout the life of a task. See “Using the interval control request token UEPICTOK” on page 148.
	UEPRECUR	Address of a usage recursion count.
Return codes	**UERCNORM**	Continue processing.
	UERCBYP	The interval control EXEC interface program should ignore this request.
	UERCPURG	Task purged during XPI call.
XPI calls	All can be used. You can also use EXEC CICS API commands at this user exit.	
	Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, you are recommended to use the EXEC CICS GETMAIN and FREEMAIN commands instead.	
API commands	All can be used.	

Warning: Care should be taken when issuing recursive commands not to cause a loop. For example, it is your responsibility to avoid entering a loop when an interval control request is issued from the XICEREQ exit.

Use of the recursion count UEPRECUR is recommended.

Notes:

1. Exit programs that issue EXEC CICS commands must first address the EIB. See the *CICS/ESA Customization Guide* for information about addressing the EIB.
2. Exit programs that issue EXEC CICS commands, and that use the DFHEIENT macro, should use the DFHEIRET macro to set a return code and return to CICS. See the *CICS/ESA Customization Guide* for information about using the DFHEIRET macro.

Exit XICEREQC

	Invoked after an interval control API request has completed, and before return from the interval control EXEC interface program.	
Exit-specific parameters	**UEPCLPS**	Address of the command-level parameter structure. See “The UEPCLPS exit-specific parameter” on page 142.
	UEPICTOK	Address of the 4 byte token to be passed to XICEREQC. This allows you, for example, to pass a work area to exit XICEREQC.
	UEPRCODE	Address of a 6-byte hexadecimal copy of the EIB return code ‘EIBRCODE’. For details of EIB return codes, refer to the *CICS/ESA Application Programming Reference* manual.
	UEPRESP	Address of a 4-byte binary copy of the EIB response code ‘EIBRESP’.
	UEPRESP2	Address of a 4-byte binary copy of the EIB response code ‘EIBRESP2’.
	UEPTSTOK	Address of a 4-byte token that is valid throughout the life of a task. See “Using the interval control request token UEPICTOK” on page 148.
	UEPRECUR	Address of a usage recursion count.
Return codes	**UERCNORM**	Continue processing.
	UERCPURG	Task purged during XPI call.
XPI calls	All can be used. Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, you are recommended to use the EXEC CICS GETMAIN and FREEMAIN commands instead.	
API commands	All can be used.	

Warning: Care should be taken when issuing recursive commands. For example, you must avoid entering a loop when issuing an interval control request from the XICEREQC exit.

Use of the recursion count UEPRECUR is recommended.

Notes:

1. Exit programs that issue EXEC CICS commands must first address the EIB. See the *CICS/ESA Customization Guide* for information about addressing the EIB.
2. Exit programs that issue EXEC CICS commands, and that use the DFHEIENT macro, should use the DFHEIRET macro to set a return code and return to CICS. See the *CICS/ESA Customization Guide* for information about using the DFHEIRET macro.

Transient data program exits XTDEREQ and XTDEREQC

The XTDEREQ exit allows you to intercept a transient data request before any action has been taken on it by transient data. The XTDEREQC exit allows you to intercept a transient data request after transient data has completed its processing.

Using XTDEREQ, you can:

- Analyze the request to determine its type, the keywords specified, and their values.
- Modify any value specified by the request before the command is executed.
- Set return codes to specify that either:
  - CICS should continue with the (possibly modified) request.
  - CICS should bypass the request. (Note that if you set this return code, you must also set up return codes for the EXEC interface block (EIB), as if you had processed the request yourself.)

Using XTDEREQC, you can:

- Analyze the request, to determine its type, the keywords specified, and their values.
- Set return codes for the EIB.

Both exits are passed six parameters as follows:

- The address of the command-level parameter structure
- The address of a token (UEPTDTOK) used to pass 4 bytes of data from XTDEREQ to XTDEREQC
- The addresses of copies of three pieces of return code information from the EIB
- The address of a token (UEPTSTOK) that is valid throughout the life of a task
- The address of an exit recursion count (UEPRECUR).

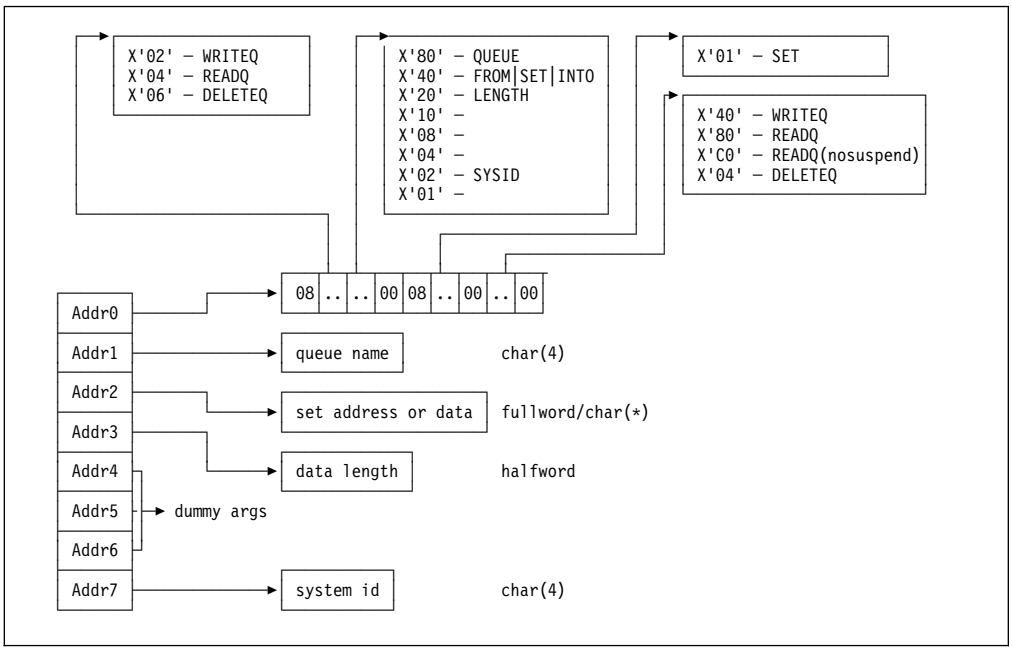


Figure 11. The command-level parameter structure for transient data

The command-level parameter structure consists of a series of addresses. The first address points to the EXEC interface descriptor (EID), which consists of an 8-byte area that describes the type of request and identifies each keyword specified with the request. The remaining addresses point to pieces of data associated with the request. (For example, the second address points to the queue name.)

You can examine the EID to determine the type of request and the keywords specified. You can examine the other parameters in the list to determine the values of the keywords. You can also modify values of keywords specified on the request. (For example, you could change the sysid specified in the request.)

End of parameter list indicator

The high-order bit is set on in the last address set in the parameter list to indicate that it is the last one in the list. On return from your user exit program, CICS scans the parameter list for the high-order bit to find the last parameter. Therefore, if you modify the length of the parameter list, you must also reset the high-order bit to indicate which is the new last address.

For example, if the parameter list specifies only the first two addresses (TD_ADDR0, the address of the EID, and TD_ADDR1, the address of the name of the queue named in a DELETEQ request), the high-order bit is set on in TD_ADDR1. If you extend the parameter list by setting address of a SYSID in TD_ADDR7, you must reset the high-order bit in TD_ADDR1 and set it on in TD_ADDR7 instead.

The maximum size of parameter list is supplied to the exit, thus allowing your exit program to add any parameters not already specified without needing to first obtain more storage.

The original parameter list, as it was before XTDEREQ was invoked, is restored after the completion of XTDEREQC. It follows that the execution diagnostic facility (EDF) displays the original command before **and** after execution. **EDF does not display any changes made by the exit**.

The UEPCLPS exit-specific parameter: The UEPCLPS exit-specific parameter is included in both exit XTDEREQ and exit XTDEREQC. It is the address of the command-level parameter structure. The command-level parameter structure contains 8 addresses, TD_ADDR0 through TD_ADDR7. These are described in the DSECT DFHTDUED, which you should copy into your exit program by including the statement COPY DFHTDUED.

The command-level parameter list is made up as follows:

TD_ADDR0
is the address of an 8-byte area called the EID, which is made up as follows:

TD_GROUP
TD_FUNCT
TD_BITS1
TD_BITS2
TD_EIDOPT5
TD_EIDOPT6
TD_EIDOPT7

TD_GROUP Always X'08', indicating that this is a transient data request.

TD_FUNCT One byte that defines the type of request:

X'02' WRITEQ
X'04' READQ
X'06' DELETEQ

TD_BITS1 Existence bits that define which arguments were specified. To obtain the argument associated with a keyword, you need to use the appropriate address from the command-level parameter structure. Before using this address, you must check the associated existence bit. If the existence bit is set off, the argument was not specified in the request and the address should not be used.

X'80' Set if the request contains an argument for the QUEUE keyword. If set, **TD_ADDR1** is meaningful.
X'40' Set if the request contains an argument for any of the INTO, SET, or FROM keywords. If set, **TD_ADDR2** is meaningful.
X'20' Set if the request contains an argument for the LENGTH keyword. If set, **TD_ADDR3** is meaningful.
X'02' Set if the request contains an argument for the SYSID keyword. If set, **TD_ADDR7** is meaningful.

TD_BITS2 Two bytes not used by transient data.

TD_EIDOPT5 Indicates whether certain keywords were specified on the request.

X'01' SET (and not INTO) was specified.

TD_EIDOPT6 One byte not used by transient data.

TD_EIDOPT7 Indicates whether certain functions and/or keywords were specified on the request:

X'40' WRITEQ specified
X'80' READQ specified
X'C0' READQ(nosuspend) specified
X'04' DELETEQ specified

TD_ADDR1
is the address of a 4-byte area containing the name from QUEUE.

TD_ADDR2
is the address of one of the following:

- A 4-byte address from SET (if the request is READQ and **TD_EIDOPT5** indicates that this is SET).
- Data from INTO (if the request is READQ and **TD_EIDOPT5** indicates that this is not SET). You cannot modify this bit in your user exit.
- Data from FROM (if the request is WRITEQ).

TD_ADDR3
is the address of one of the following:

- The halfword value of LENGTH (if the request is READQ or WRITEQ). **Warning:** For requests that specify INTO, do not change the value of LENGTH to a value greater than that specified by the application. To do so causes a storage overlay in the application.

TD_ADDR4
is the address of a value intended for CICS internal use only. It must not be used.

TD_ADDR5
is the address of a value intended for CICS internal use only. It must not be used.

TD_ADDR6
is the address of a value intended for CICS internal use only. It must not be used.

TD_ADDR7
is the address of an area containing the value of SYSID.

TD_ADDR8
is the address of a value intended for CICS internal use only. It must not be used.

Modifying fields in the command-level parameter structure: Some fields that are passed to transient data are used as input to the request, some are used as output fields, and some are used for both input and output. The method your user exit program uses to modify a field depends on the usage of the field.

The following are always input fields:

QUEUE
FROM
SYSID

The following are always output fields:

INTO
SET

LENGTH is an input field on a WRITEQ request, and an output field on a READQ request that specifies SET. It is both an input and an output field on a READQ request that specifies INTO.

RTRANSID
RTERMID
QUEUE

Modifying input fields: The correct method of modifying an input field is to create a new copy of it, and to change the address in the command-level parameter list to point to your new data. **Warning:** You must never modify an input field by altering the data that is pointed to by the command-level parameter list. To do so would corrupt storage belonging to the application program and would cause a failure when the program attempted to reuse the field.

Modifying output fields: The technique described in "Modifying input fields" is not suitable for modifying output fields. (The results would be returned to the new area instead of the application's area, and would be invisible to the application.)

An output field is modified by altering the data that is pointed to by the command-level parameter list. In the case of an output field, you can modify the application's data in place, because the application is expecting the field to be modified.

Modifying fields used for both input and output: An example of a field that is used for both input and output is LENGTH on a READQ request that specifies INTO. You can treat such fields in the same way as output fields, and they are considered to be the same.

Modifying the EID: It is not possible to modify the EID to make major changes to requests, such as changing a READQ request to a WRITEQ request.

However, you can make minor changes to requests, such as turning on the existence bit for SYSID so that the request can be changed into one that is shipped to a remote system.

The list that follows shows the bits in the EID that **can** be modified. Any attempt to modify any other part of the EID is ignored.

TD_BITS1

X'20' The existence bit for LENGTH.
X'02' The existence bit for SYSID.

TD_EIDOPT5

X'01' Existence bit for SET keyword. You cannot modify this bit from your user exit.

TD_EIDOPT7

Changes to TD_EIDOPT7 are limited to READQ requests. X'80'–READQ is interchangeable with X'C0'–READQ(nosuspend). No other changes may be made to this byte.

Your exit program may provide its own command-level parameter structure and EID. If it provides its own command-level parameter structure, you should modify UEPCLPS to point to the new structures. If it provides its own EID, you should modify TD_ADDR0 to point to the new structures.

The EID is reset to its original value before return to the application program. That is, changes made to the EID are retained for the duration of the transient data request only. **Warning:** Your user exit program is prevented from making major changes to the EID. However, you must take great care when making the minor modifications that **are** permitted.

Use of the task token UEPTSTOK: UEPTSTOK provides the address of a 4-byte area that you can use to pass information between successive transient data requests in the same task. (By contrast, UEPTDTOK is usable only for the duration of a single transient data request, because its contents may be destroyed at the end of the request.) For example, if you need to pass information between successive invocations of the XTDEREQ exit, UEPTSTOK provides a means of doing this.

The EIB: Copies of EIBRCODE, EIBRESP, and EIBRESP2 are passed to the exit, so that you can:

- Modify or set completion information in XTDEREQ and XTDEREQC
- Examine completion information in XTDEREQC.

You can update the copies of EIBRCODE, EIBRESP, and EIBRESP2 that you are given in the parameter list. Transient data copies your values into the real EIB after the completion of XTDEREQC; or if you specify a return code of ' bypass' in XTDEREQ.

You must set valid transient data responses. You must set all three of EIBRCODE, EIBRESP, and EIBRESP2 to a consistent set of values, such as would be set by CICS transient data to describe a valid completion. **CICS does not police the consistency of EIBRCODE, EIBRESP, and EIBRESP2**. However, if EIBRCODE is set to a non–zero value and EIBRESP is set to zero then CICS will override EIBRESP with a non–zero value. To aid you in setting the values of EIBRCODE, EIBRESP, and EIBRESP2, the values used by transient data are specified in DFHTDUED.

Exit XTDEREQ

	Invoked before CICS processes a transient data API request.	
Exit-specific parameters	**UEPCLPS**	Address of the command-level parameter structure. See “The UEPCLPS exit-specific parameter” on page 154.
	UEPTDTOK	Address of the 4-byte token to be passed to XTDEREQC. This allows you, for example, to pass a work area to exit XTDEREQC.
	UEPRCODE	Address of a 6-byte hexadecimal copy of the EIB return code ‘EIBRCODE’. For details of EIB return codes, refer to the *CICS/ESA Application Programming Reference* manual.
	UEPRESP	Address of a 4-byte binary copy of the EIB response code ‘EIBRESP’.
	UEPRESP2	Address of a 4-byte binary copy of the EIB response code ‘EIBRESP2’.
	UEPTSTOK	Address of a 4-byte token that is valid throughout the life of a task. See “Use of the task token UEPTSTOK.”
	UEPRECUR	Address of a usage recursion count.
Return codes	**UERCNORM**	Continue processing.
	UERCBYP	The transient data EXEC interface program should ignore this request.
	UERCPURG	Task purged during XPI call.
XPI calls	All can be used.	
	Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, we recommend that you use the EXEC CICS GETMAIN and FREEMAIN commands instead.	
API commands	All can be used.	

Warning: Care should be taken when issuing recursive commands. For example, you must avoid entering a loop when issuing a transient data request from the XTDEREQ exit.

Use of the recursion count UEPRECUR is recommended.

Notes:

1. Exit programs that issue EXEC CICS commands must first address the EIB. See the *CICS/ESA Customization Guide* for information about addressing the EIB.
2. Exit programs that issue EXEC CICS commands, and that use the DFHEIENT macro, should use the DFHEIRET macro to set a return code and return to CICS. See the *CICS/ESA Customization Guide* for information about using the DFHEIRET macro.

Exit XTDEREQC

	Invoked after a transient data API request has completed, and before return from the transient data EXEC interface program.	
Exit-specific parameters	**UEPCLPS**	Address of the command-level parameter structure. See “The UEPCLPS exit-specific parameter” on page 154.
	UEPTDTOK	Address of the 4 byte token to be passed to XTDEREQC. This allows you, for example, to pass a work area to exit XTDEREQC.
	UEPRCODE	Address of a 6-byte hexadecimal copy of the EIB return code ‘EIBRCODE’. For details of EIB return codes, refer to the *CICS/ESA Application Programming Reference* manual.
	UEPRESP	Address of a 4-byte binary copy of the EIB response code ‘EIBRESP’.
	UEPRESP2	Address of a 4-byte binary copy of the EIB response code ‘EIBRESP2’.
	UEPTSTOK	Address of a 4-byte token that is valid throughout the life of a task. See “Use of the task token UEPTSTOK” on page 157.
	UEPRECUR	Address of a usage recursion count.
Return codes	**UERCNORM**	Continue processing.
	UERCPURG	Task purged during XPI call.
XPI calls	All can be used. Although the exit permits the use of XPI GETMAIN and FREEMAIN calls, we recommend that you use the EXEC CICS GETMAIN and FREEMAIN commands instead.	
API commands	All can be used.	

Warning: Care should be taken when issuing recursive commands. For example, you must avoid entering a loop when issuing a transient data request from the XTDEREQC exit.

Use of the recursion count UEPRECUR is recommended.

Notes:

1. Exit programs that issue EXEC CICS commands must first address the EIB. See the *CICS/ESA Customization Guide* for information about addressing the EIB.
2. Exit programs that issue EXEC CICS commands, and that use the DFHEIENT macro, should use the DFHEIRET macro to set a return code and return to CICS. See the *CICS/ESA Customization Guide* for information about using the DFHEIRET macro.

Temporary storage program exits XTSEREQ and XTSEREQC

The XTSEREQ exit allows you to intercept temporary storage API requests before any action has been taken on the request. The XTSEREQC exit allows you to intercept the response after a temporary storage API request has completed.

The EXEC CICS API requests affected are:

- WRITEQ TS
- READQ TS
- DELETEQ TS

Using **XTSEREQ**, you can:

- Analyze the API parameter list (function, keywords, argument values, and responses)
- Modify any input parameter value prior to execution of a request
- Prevent execution of a request.

Using **XTSEREQC**, you can:

- Analyze the API parameter list
- Modify any output parameter value after request completion.

You can also:

- Pass data between your XTSEREQ and XTSEREQC exit programs when they are invoked for the same request
- Pass data between your temporary storage exit programs when they are invoked within the same task.

The temporary storage user exits from earlier releases of CICS (XTSOUT, XTSIN, and XTSREQ) are not affected by these changes. However, it is possible that the existing exit points can modify situations set up by XTSEREQC, and therefore you must consider the order in which the exits are invoked.

If all five exit points are enabled, the order of invocation is as follows:

- For the WRITEQ temporary storage command:
  1. XTSEREQ
  2. XTSREQ
  3. XTSOUT
  4. XTSEREQC
- For the READQ temporary storage command:
  1. XTSEREQ
  2. XTSREQ
  3. XTSIN
  4. XTSEREQC
- For the DELETEQ temporary storage command:
  1. XTSEREQ
  2. XTSREQ
  3. XTSEREQC

Exit XTSEREQ

	Invoked before CICS processes a temporary storage API request.	
Exit-specific parameters	**UEPCLPS**	Address of a copy of the command parameter list. See "The command-level parameter structure" on page 162.
	UEPTQTOK	Address of a 4-byte area which can be used to pass information between XTSEREQ and XTSEREQC for a single temporary storage request.
	UEPRCODE	Address of a 6-byte hexadecimal copy of the EIB return code EIBRCODE. For details of EIB return codes, see the *CICS/ESA Application Programming Reference* manual.
	UEPRESP	Address of a 4-byte binary copy of the EIB response code EIBRESP.
	UEPRESP2	Address of a 4-byte binary copy of the EIB response code EIBRESP2.
	UEPTSTOK	Address of a 4-byte token which can be used to pass information between successive temporary storage requests within the same task (for example, between successive invocations of the XTSEREQ exit).
	UEPRECUR	Address of a usage recursion count.
Return codes	**UERCBYP**	Bypass this request
	UERCNORM	Continue processing.
	UERCPURG	Task purged during XPI call.
XPI calls	All can be used.	
API commands	All can be used.	

Warning: Care should be taken when issuing recursive commands. For example, you must avoid entering a loop when issuing a temporary storage request from the XTSEREQ exit.

Use of the recursion count UEPRECUR is recommended.

Exit XTSEREQC

	Invoked after a temporary storage API request has completed, before return from the temporary storage EXEC interface program.	
Exit-specific parameters	**UEPCLPS**	Address of a copy of the command parameter list. See "The command-level parameter structure" on page 162.
	UEPTQTOK	Address of a 4-byte area which can be used to pass information between XTSEREQ and XTSEREQC for a single temporary storage request.
	UEPRCODE	Address of a 6-byte hexadecimal copy of the EIB return code EIBRCODE. For details of EIB return codes, see the *CICS/ESA Application Programming Reference* manual.
	UEPRESP	Address of a 4-byte binary copy of the EIB response code EIBRESP.
	UEPRESP2	Address of a 4-byte binary copy of the EIB response code EIBRESP2.
	UEPTSTOK	Address of a 4-byte token which can be used to pass information between successive temporary storage requests within the same task (for example, between successive invocations of the XTSEREQC exit).
	UEPRECUR	Address of a usage recursion count.
Return codes	**UERCNORM**	Continue processing.
	UERCPURG	Task purged during XPI call.
XPI calls	All can be used.	
API commands	All can be used.	

Warning: Care should be taken when issuing recursive commands not to cause a loop. For example, it is your responsibility to avoid entering a loop when issuing a temporary storage request from the XTSEREQC exit.

Use of the recursion count UEPRECUR is recommended.

The command-level parameter structure: The command-level parameter structure consists of a series of addresses, which are passed to a user exit program invoked at either XTSEREQ or XTSEREQC. These addresses are:

UEPCLPS
address of a copy of the command parameter list.

The command parameter list for a temporary storage command has the arguments listed in "The UEPCLPS exit-specific parameter" on page 163.

UEPTQTOK
address of a 4-byte area which can be used to pass information between XTSEREQ and XTSEREQC for a single temporary storage request.

UEPRCODE, UEPRESP, UEPRESP2
addresses of copies of EIBRCODE, EIBRESP, and EIBRESP2.

This enables user exit programs to:

- Modify or set completion information in XTSEREQ and XTSEREQC
- Examine completion information in XTSEREQC.

You can update the copies of EIBRCODE, EIBRESP, and EIBRESP2 that you are given in the parameter list. If you update the values, temporary storage copies the new values into the application program's EIB after the completion of XTSEREQC or if you specify a return code of 'bypass' in XTSEREQ.

You must set valid temporary storage responses. You must set all three of EIBRCODE, EIBRESP, and EIBRESP2 to a consistent set of values, such as would be set by temporary storage to describe a valid completion. CICS does not check the consistency of EIBRCODE, EIBRESP, and EIBRESP2. If EIBRCODE is set to a non-zero value and EIBRESP is set to zero, CICS will override EIBRESP with a non-zero value. To help you set values for EIBRCODE, EIBRESP, and EIBRESP2, the values used by temporary storage are specified in DSECT DFHTSUED.

UEPTSTOK
address of a 4-byte token which can be used to pass information between successive temporary storage requests within the same task. For example, you can use UEPTSTOK to pass information between successive invocations of the XTSEREQ exit.

UEPRECUR
address of the usage recursion count.

The UEPCLPS exit-specific parameter: The UEPCLPS exit-specific parameter is included in both exit XTSEREQ and exit XTSEREQC. It is the address of the command-level parameter structure. The command-level parameter structure contains 8 addresses, TS_ADDR0 through TS_ADDR7. These are described in the DSECT DFHTSUED, which you should copy into your exit program by including the statement COPY DFHTSUED.

The command-level parameter list is made up as follows.

Note: The relationship between arguments, keywords, data types, and input/output types is summarized for the temporary storage commands in the following tables:

Command	See
WRITEQ TS	Table 8 on page 167
READQ TS	Table 9 on page 167
DELETEQ TS	Table 10 on page 168

TS_ADDR0
is the address of an 8-byte area called the EID, which is made up as follows:

TS_GROUP
TS_FUNCT
TS_BITS1
TS_BITS2
TS_EIDOPT5

TS_EIDOPT6
TS_EIDOPT7

TS_GROUP Always X'0A', indicating that this is a temporary storage request.

TS_FUNCT One byte that defines the type of request:

X'02' WRITEQ
X'04' READQ
X'06' DELETEQ

TS_BITS1 Existence bits that define which arguments were specified. To obtain the argument associated with a keyword, you need to use the appropriate address from the command-level parameter structure. Before using this address, you must check the associated existence bit. If the existence bit is set off, the argument was not specified in the request and the address should not be used.

X'80' Set if the request contains an argument for the QUEUE keyword. If set, **TS_ADDR1** is meaningful.
X'40' Set if the request contains an argument for any of the FROM, INTO, or SET keywords. If set, **TS_ADDR2** is meaningful.
X'20' Set if the request contains an argument for the LENGTH keyword. If set, **TS_ADDR3** is meaningful.
X'10' Set if the request contains an argument for the NUMITEMS keyword. If set, **TS_ADDR4** is meaningful.
X'08' Set if the request contains an argument for the NUMITEMS or ITEM keyword. If set, **TS_ADDR5** is meaningful.
X'02' Set if the request contains an argument for the SYSID keyword. If set, **TS_ADDR7** is meaningful.

TS_BITS2 Two bytes not used by temporary storage.

TS_EIDOPT5 Indicates whether certain keywords were specified on the request.

X'01' SET was specified (otherwise INTO). You cannot modify this bit from your user exit.

TS_EIDOPT6 One byte not used by temporary storage.

TS_EIDOPT7 Indicates whether certain functions and/or keywords were specified on the request.

X'10' WRITEQ NOSUSPEND specified.
X'80' WRITEQ MAIN or READQ ITEM specified.
X'04' WRITEQ REWRITE or READQ NUMITEMS specified.

TS_EIDOPT8 Indicates whether certain keywords were specified on the request.

X'80' ITEM was specified (otherwise NUMITEMS).

TS_ADDR1
is the address of an 8-byte area containing the name from QUEUE.

TS_ADDR2
is the address of one of the following:

- A 4-byte address from SET (if the request is READQ and **TS_EIDOPT5** indicates that this is SET).
- Data from INTO (if the request is READQ and **TS_EIDOPT5** indicates that this is not SET).
- Data from FROM (if the request is WRITEQ).

TS_ADDR3
is the address of the halfword value of LENGTH (if the request is READQ or WRITEQ).

Warning: For requests that specify INTO, do not change the value of LENGTH to a value greater than that specified by the application. To do so causes a storage overlay in the application.

TS_ADDR4
is the address of the halfword value of NUMITEMS (if the request is READQ).

TS_ADDR5
is the address of one of the following:

- The halfword value of NUMITEMS (if the request is WRITEQ).
- The halfword value of ITEM (if the request is READQ or WRITEQ).

TS_ADDR6
is the address of a value intended for CICS internal use only. It must not be used.

TS_ADDR7
is the address of an area containing the value of SYSID.

Modifying fields in the command-level parameter structure: Some fields that are passed to temporary storage are used as input to the request, some are used as output fields, and some are used for both input and output. The method your user exit program uses to modify a field depends on the usage of the field.

The following are always input fields:

QUEUE
FROM
SYSID

The following are always output fields:

INTO
NUMITEMS
SET

LENGTH is an input field on a WRITEQ request, and an output field on a READQ request that specifies SET. It is both an input and an output field on a READQ request that specifies INTO.

ITEM is an input field on a READQ request, and on a WRITEQ request that specifies REWRITE. It is both an input and an output field on a WRITEQ request that does not specify REWRITE.

Modifying input fields: The correct method of modifying an input field is to create a new copy of it, and to change the address in the command-level parameter list to point to your new data. **Warning:** You must never modify an input field by altering the data that is pointed to by the command-level parameter list. To do so would corrupt storage belonging to the application program and would cause a failure when the program attempted to reuse the field.

Modifying output fields: The technique described in "Modifying input fields" is not suitable for modifying output fields. (The results would be returned to the new area instead of the application's area, and would be invisible to the application.)

An output field is modified by altering the data that is pointed to by the command-level parameter list. In the case of an output field, you can modify the application's data in place, because the application is expecting the field to be modified anyway.

Modifying fields used for both input and output: An example of a field that is used for both input and output is LENGTH on a READQ request that specifies INTO. You can treat such fields in the same way as output fields, and they are considered to be the same.

Modifying the EID: It is not possible to modify the EID to make major changes to requests. It is not possible, for example, to change a READQ request to a WRITEQ request.

However, you can make minor changes to requests, such as to turn on the existence bit for SYSID so that the request can be changed into one that is shipped to a remote system.

The list that follows shows the bits in the EID that can be modified. Any attempt to modify any other part of the EID is ignored.

TS_BITS1

X'02' The existence bit for SYSID.

TS_EIDOPT7

A user exit program at XTSEREQ can set the following on or off for all WRITEQ TS commands:

X'10' The existence bit for NOSUSPEND.
X'08' The existence bit for MAIN.

Your exit program may provide its own command-level parameter structure and EID, in which case you should modify UEPCLPS and TS_ADDR0 respectively to point to the new structures.

The EID is reset to its original value before return to the application program. That is, changes made to the EID are retained for the duration of the temporary storage request only. **Warning:** Your user exit program is prevented from making major changes to the EID. However, you must take great care when making the minor modifications that **are** permitted.

Use of the task token UEPTSTOK: UEPTSTOK provides the address of a 4-byte area that you can use to pass information between successive temporary storage requests in the same task. (By contrast, UEPTQTOK is usable only for the duration of a single temporary storage request, because its contents may be destroyed at the end of the request.) For example, if you need to pass information between successive invocations of the XTSEREQ exit, UEPTSTOK provides a means of doing this.

Table 8. WRITEQ TS: User arguments and associated keywords, data types, and input/output types

Argument	Keyword	Data type	Input/output type
Arg1	QUEUE	CHAR(8)	input
Arg2	FROM	DATA-AREA	input
Arg3	LENGTH	BIN(15)	input
Arg4	*	*	*
Arg5	ITEM	BIN(15)	input/output
Arg5	NUMITEMS	BIN(15)	output
Arg6	*	*	*
Arg7	SYSID	CHAR(4)	input

Note: The different uses of Arg5 are shown, because Arg5 is used by the ITEM and NUMITEMS keywords which are alternatives and the argument to the ITEM keyword is an input field when REWRITE is specified.

Table 9. READQ TS: User arguments and associated keywords, data types, and input/output types

Argument	Keyword	Data type	Input/output type
Arg1	QUEUE	CHAR(8)	input
Arg2	SET	DATA-AREA, PTR	output
Arg2	INTO	DATA-AREA	output
Arg3	LENGTH	BIN(15)	input/output
Arg4	NUMITEMS	BIN(15)	output
Arg5	ITEM	BIN(15)	input
Arg6	*	*	
Arg7	SYSID	CHAR(4)	input

Table 10. DELETEQ TS: User arguments and associated keywords, data types, and input/output types			
Argument	**Keyword**	**Data type**	**Input/output type**
Arg1	QUEUE	CHAR(8)	input
Arg2	*	*	*
Arg3	*	*	*
Arg4	*	*	*
Arg5	*	*	*
Arg6	*	*	*
Arg7	SYSID	CHAR(4)	input

Modifying user arguments: User exit programs can modify user arguments, as follows:

For input arguments, the user exit program should obtain sufficient storage to hold the modified argument, set up that storage to the required value, and set the associated pointer in the parameter list to the address of the newly acquired area.

For output arguments, and for input/output arguments, the user exit program can update the argument in place, because the area of storage is represented by a variable in the application which is expected to receive a value from CICS.

Notes:

1. CICS does not check changes to argument values, so any changes must be verified by the user exit program making the changes.
2. It is not advisable for XTSEREQ to modify output arguments or for XTSEREQC to modify input arguments.

Adding user arguments: Global user exit programs can add arguments associated with the SYSID keyword. You must ensure that the arguments you specify or modify in your exit programs are valid.

Assuming that the argument to be added does not already exist, the user exit program must:

1. Obtain storage for the argument to be added
2. Initialize the storage to the required value
3. Select and set up the appropriate pointer from the parameter list
4. Select and set up the appropriate argument existence bit in Arg0
5. Modify the parameter list to reflect the new end of list indicator.

Removing user arguments: User exit programs can remove arguments (for which the program is totally responsible) associated with the SYSID keyword:

Assuming that the argument to be removed exists, the user exit program must:

1. Switch the corresponding argument existence bit to '0'b in Arg0
2. Modify the parameter list to reflect the new end of list indicator.

Chapter 11. External CICS interface

This chapter describes the external CICS interface (EXCI) introduced in CICS/ESA 4.1. It covers the following topics:

- Overview
- Benefits of the external CICS interface
- Requirements for the external CICS interface
- Changes to CICS externals
- Problem determination
- Choosing between the EXEC CICS LINK or CALL interface
- Compiling and link-editing external CICS interface client programs
- Sample application programs.

It also describes some sample client and server application programs that use the external CICS interface and provides advice on how to install and run them.

Overview

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/ESA 4.1 region and to pass and receive data by means of a communications area. The CICS program is invoked as if linked-to by another CICS program.

This programming interface allows a user to allocate and open sessions (or pipes[6],) to a CICS system and to pass distributed program link (DPL) requests over them. The CICS interregion communication program (DFHIRP) supports these DPL requests from a non-CICS client program.

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/ESA 4.1. (See Chapter 5, “Cross-system multiregion operation (XCF/MRO)” on page 87 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.

---

6 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.

Benefits of the external CICS interface

CICS applications are now more easily accessible from non-CICS environments.

Programs running in MVS can now issue the EXEC CICS LINK PROGRAM command to call application programs running in CICS/ESA 4.1 regions. Alternatively, the MVS programs can use the CALL interface when it is more appropriate to do so.

The provision of this new programming interface means that, typically, an MVS batch job can:

- Update resources with integrity while CICS is accessing them.
- Take CICS resources offline (and back online) at the start (and end) of the batch job. For example, you can:
  - Open and close CICS files.
  - Enable and disable transactions in CICS (and so eliminate the need for a master terminal operator during system backup and recovery procedures).

Distributed computing environment

Although the CICS external interface operates over CICS MRO links, the client program can run on non-MVS platforms, and pass the requests over an open system interface (OSI) using OpenEdition DCE Base Services Feature. Thus the external CICS interface provides an open interface to a wide variety of other application platforms.

Requirements for the external CICS interface

Client programs running in an MVS address space can communicate only with CICS server regions running under CICS/ESA 4.1 or a later, upward-compatible release. This is because of the changes to the MRO connection definition to support the external CICS interface.

Also, the client program can connect to the server CICS region only through the CICS/ESA 4.1, or later, interregion communication program, DFHIRP.

Changes to CICS externals

The introduction of the external CICS interface in CICS/ESA 4.1 results in a number of changes to CICS externals. These are:

- Changes to the CICS system programming interface (SPI)
- The CALL application programming interface
- Changes to the CICS application programming interface (API)
- Changes to resource definition
- Changes to user replaceable modules
- External CICS interface options table, DFHXCOPT.

Changes to the system programming interface

The CICS system programming interface command, EXEC CICS INQUIRE CONNECTION, is extended to support the external CICS interface.

EXEC CICS INQUIRE CONNECTION command

General-use programming interface

The EXEC CICS INQUIRE CONNECTION command is expanded as follows:

- There are three new keywords:
  - CONNTYPE
  - RECEIVECOUNT
  - SENDCOUNT
- There is an extra CVDA value is provided on the PROTOCOL keyword.

These additions allow an application program to determine whether the external CICS interface connection is:

- For use by multiple users (it is generic), or
- Dedicated for use by a single user (it is specific).

The additions also enable you to determine, via RECEIVECOUNT, the number of receive sessions defined for this connection definition. The number of receive sessions equates to the number of sessions, or pipes that can be opened between the non-CICS program and the CICS system.

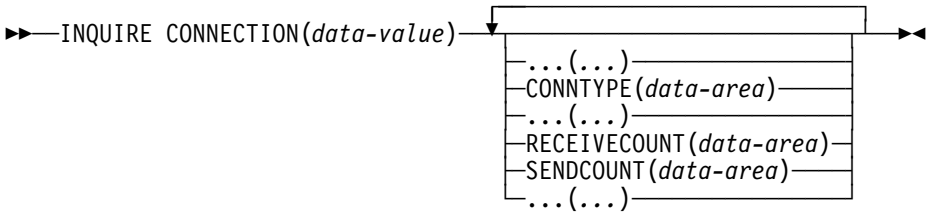
The number of send sessions defined for this connection definition is returned in SENDCOUNT.

Additionally, a new CVDA value returned on the PROTOCOL keyword indicates whether this is an external CICS interface connection.

Function

Retrieve information about a named connection definition.

Syntax



Conditions:
NOTAUTH, SYSIDERR, ILLOGIC

INQUIRE CONNECTION options

CONNTYPE(cvda)
returns a CVDA value identifying the type of external CICS interface (EXCI) connection. CVDA values are:

GENERIC
This is an EXCI connection, with many sessions to be shared by multiple users.

SPECIFIC
This is an EXCI connection, with one or more sessions dedicated to a single user.

NOTAPPLIC
The connection is used for CICS-to-CICS MRO or it is an indirect connection.

PROTOCOL(cvda)
returns a CVDA value identifying the protocol in use if this is a VTAM or external CICS interface connection. CVDA values are:

APPC
The connection uses the VTAM LUTYPE6.2 protocol for intersystem communication.

EXCI
The connection uses the external CICS interface for communication between CICS and non-CICS systems.

LU61
The connection uses the VTAM LUTYPE6.1 protocol for CICS-to-CICS or CICS-to-IMS intersystem communication.

NOTAPPLIC
The connection is used for CICS-to-CICS MRO or it is INDIRECT

RECEIVECOUNT(data-area) (MRO and EXCI connections only)
returns a fullword binary string indicating the number of RECEIVE sessions defined for this connection.

If the connection is not an MRO or EXCI connection, CICS returns a value of –1.

SENDCOUNT(data-area) (MRO and EXCI connections only)
returns a fullword binary string indicating the number of send sessions defined for this connection.

If the connection is not an MRO or EXCI connection, CICS returns a value of –1.

The other options of the EXEC CICS INQUIRE CONNECTION command are not changed by the external CICS interface. For the full syntax of the command, and details of changes resulting from changes to other functions, see Chapter 35, “Changes to the system programming interface (SPI)” on page 585.

The CALL application programming interface

The external CICS interface provides two forms of programming interface: the external CICS CALL interface and an EXEC CICS LINK PROGRAM command.

The EXCI CALL interface consists of six commands that allow you to:

- Allocate and open sessions to a CICS system from non-CICS programs running under MVS/ESA, and
- To issue distributed program link (DPL) requests on these sessions from the non-CICS programs.

The six EXCI commands are:

- Initialize_User
- Allocate_Pipe
- Open_Pipe
- DPL_Request
- Close_Pipe
- Deallocate_Pipe

The application program stub, DFHXCSTB

The EXCI commands invoke the external CICS interface via an application programming stub provided by CICS, called DFHXCSTB. You must include this stub when you link-edit your non-CICS program.

The CALL interface commands

In the description of each command that follows, the syntax box illustrates the assembler form of the command. The commands are also supported by the C, COBOL II and PL/I programming languages, using the CALL conventions appropriate for the language.

There are examples of these CALLs, in all the supported languages, in the sample program client programs provided by CICS. See “Sample application programs” on page 211 for information about these.

INITIALIZE_USER

Function

Initialize the user environment, including obtaining authority to use IRC facilities. The environment is created for the lifetime of the TCB, so the command needs to be issued only once per user per TCB. Further commands from this user must be issued under the same TCB.

Note: A *user* is a program that has issued an Initialize_user request (or for which an Initialize_user request has been issued), with a unique name per TCB. For example:

- A simple client program running under MVS can be a single user of the external CICS interface.
- A client program running under MVS can open several pipes and issue external CICS interface calls over them sequentially, on behalf of different vendor packages. In this case, from the viewpoint of the client program, each of the packages is a user, identified by a unique user name. Thus a single client program can operate on behalf of multiple users.
- A program running under MVS can attach several TCBs, under each of which a vendor package issues external CICS interface calls on its own behalf. Each package is a client program in its own right, and runs under its own TCB. Each is also a user, with a unique user name.

Syntax

```
►►─CALL DFHXCIS,(version_number,─return_area,─user_token,─call_type,──────────►
►─user_name),VL,MF=(E,(1))─►◄
```

Initialize_User command parameters

version_number

A full-word binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See page 192 for copybook details.

return_area

A 5-word output area to receive response and reason codes, and a message pointer field. For more details see “Return area for the EXCI CALL interface” on page 192.

user_token

A 1-word output area containing a 32-bit token supplied by the CICS external interface to represent the client program.

The user token corresponds to the *user-name* parameter. The client program must pass this token on all subsequent external CICS interface commands made for the user defined on the *user_name* parameter.

call_type

A 1-word input area indicating the function of the command. It must be set to 1 in the client program to indicate that this is an Initialize_User command.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is INIT_USER. See page 192 for copybook details.

user_name

An input area holding a name that identifies the user of the external CICS interface. Generally, this is the client program. If this user is to use a specific pipe, then the value in *user_name* must match that of the NETNAME attribute of the CONNECTION definition for the specific pipe.

Response and reason codes on the Initialize_User call

For all non-zero response codes, a unique reason code value identifies the reason for the response.

The following is a summary of the response and reason codes that the external CICS interface can return on the Initialize_User call:

Response OK:

0 Normal response

Response WARNING:

3 VERIFY_BLOCK_FM_ERROR
4 WS_FREEMAIN_ERROR

Response RETRYABLE:

201 NO_CICS_IRC_STARTED

Response USER_ERROR:

401 INVALID_CALL_TYPE
402 INVALID_VERSION_NUMBER
403 INVALID_USER_NAME
410 DFHMEBM_LOAD_FAILED
411 DFHMET4E_LOAD_FAILED
412 DFHXCURM_LOAD_FAILED
413 DFHXCTRA_LOAD_FAILED
419 CICS_AFCB_PRESENT
420 DFHXCOPT_LOAD_FAILED
421 RUNNING_UNDER_AN_IRB

Response SYSTEM_ERROR:

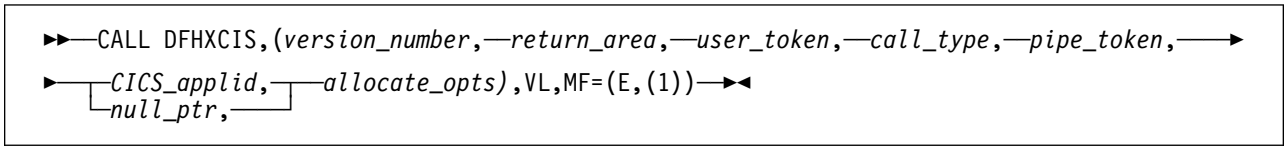
601 WS_GETMAIN_ERROR
602 XCGLOBAL_GETMAIN_ERROR
603 XCUSER_GETMAIN_ERROR
605 VERIFY_BLOCK_GM_ERROR
606 SSI_VERIFY_FAILED
607 CICS_SVC_CALL_FAILURE
622 ESTAE_SETUP_FAILURE
623 ESTAE_INVOKED
627 INCORRECT_SVC_LEVEL

ALLOCATE_PIPE

Function

Allocate a single session, or pipe, to a CICS region. This command does not connect the client program to a CICS region; this happens on the Open_Pipe command.

Syntax



Allocate_Pipe command parameters

version_number

A full-word binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See page 192 for copybook details.

return_area

A 5-word output area to receive response and reason codes, and a message pointer field. For more details see “Return area for the EXCI CALL interface” on page 192.

user_token

The 1-word token returned on the Initialize_User command.

call_type

A 1-word input area indicating the function of the command. It must be set to 2 in the client program to indicate that this is an Allocate_Pipe command.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is ALLOCATE_PIPE. See page 192 for copybook details.

pipe_token

A 1-word output area. CICS returns a 32-bit token in this area to represent the allocated session. This token must be used on any subsequent command that uses this session.

CICS_applid

An 8-byte input area containing the generic applid of the CICS system to which the allocated session is to be connected.

Although an applid is required to complete the Allocate_Pipe function, this parameter is optional on the Allocate_Pipe call. You can either specify the applid on this parameter to the Allocate_Pipe call, or in the user-replaceable module, DFHXCURM, using the URMAPPL parameter (DFHXCURM is always invoked during Allocate_Pipe processing). You can also use the URMAPPL parameter in DFHXCURM to override an applid specified on the Allocate_Pipe call. See “The external CICS interface user-replaceable module, DFHXCURM” on page 202 for information about the URMAPPL parameter.

If you omit the applid from the call, you must ensure that the CALL parameter list contains a null address for *CICS_applid*. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see “Example of EXCI CALLs with null parameters” on page 193.

allocate_opts

A 1-byte input area to represent options specified on this command. The options specify which type of session is to be used—specific or generic. `X'00'` represents a specific session. `X'80'` represents a generic session.

The equated value for these options in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) are SPECIFIC_PIPE and GENERIC_PIPE. See page 192 for copybook details.

Response and reason codes on the Allocate_Pipe call

For all non-zero response codes, a unique reason code value identifies the reason for the response.

The following is a summary of the response and reason codes that the external CICS interface can return on the Allocate_Pipe call:

Response OK:

0 Normal response

Response USER_ERROR:

401 INVALID_CALL_TYPE
402 INVALID_VERSION_NUMBER
404 INVALID_USER_TOKEN
421 RUNNING_UNDER_AN_IRB

Response SYSTEM_ERROR:

604 XCPIPE_GETMAIN_ERROR
608 IRC_LOGON_FAILURE
622 ESTAE_SETUP_FAILURE
623 ESTAE_INVOKED
628 IRP_LEVEL_CHECK_FAILURE

OPEN_PIPE

Function

Cause IRC to connect an allocated pipe to a receive session of the appropriate connection defined in the CICS region named either on the Allocate_Pipe command, or in DFHXCURM. The appropriate connection is either:

- The EXCI connection with a NETNAME value equal to the *user_name* parameter on the Initialize_User command (that is, you are using a specific connection, dedicated to this client program),

or

- The EXCI connection defined as generic.

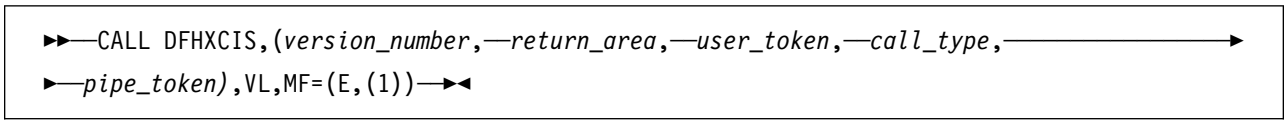
Note: This command should be used only when there is a DPL call ready to be issued to the CICS system. When not in use, EXCI sessions should not be left open.

If sessions are left open, CICS may not be able to shut its IRC facility in an orderly manner. A normal shutdown of CICS waits if any EXCI sessions are not closed. CICS issues message DFHIR2321 indicating the following information:

- The netname of the session if it is on a specific connection
- The word GENERIC if the open sessions are on a generic connection.

Provided that at least one DPL_Request call has been issued on the session, message DFHIR2321 also shows the job name, step name, and procedure name of the client job that is using the session, and the MVSID of the MVS image on which the client program is running.

Syntax



Open_Pipe command parameters

version_number

A full-word binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See page 192 for copybook details.

return_area

A 5-word output area to receive response and reason codes, and a message pointer field. For more details, see “Return area for the EXCI CALL interface” on page 192.

user_token

The 1-word token returned on the Initialize_User command

call_type

A 1-word input area indicating the function of the command. This must be set to 3 in the client program to indicate that this is an Open_pipe command.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is OPEN_PIPE. See page 192 for copybook details.

pipe_token
A 1-word output area containing the token passed by CICS on the Allocate_Pipe command. It represents the pipe being opened on this command.

Response and reason codes on the Open_Pipe call

For all non-zero response codes, a unique reason code value identifies the reason for the response.

The following is a summary of the response and reason codes that the external CICS interface can return on the Open_Pipe call:

Response OK:

0 NORMAL

Response WARNING:

1 PIPE_ALREADY_OPEN

Response RETRYABLE:

202 NO_PIPE
203 NO_CICS

Response USER_ERROR:

401 INVALID_CALL_TYPE
402 INVALID_VERSION_NUMBER
404 INVALID_USER_TOKEN
418 INVALID_PIPE_TOKEN
421 RUNNING_UNDER_AN_IRB

Response SYSTEM_ERROR:

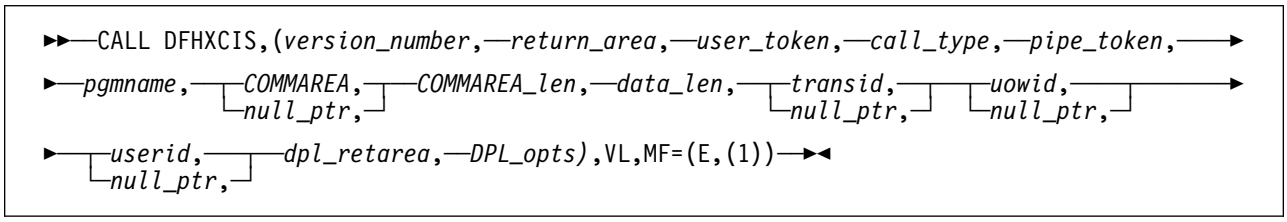
609 IRC_CONNECT_FAILURE
621 PIPE_RECOVERY_FAILURE
622 ESTAE_SETUP_FAILURE
623 ESTAE_INVOKED

DPL_Request

Function

Issue a distributed program link request across an open pipe connected to the CICS system on which the server (or target) application program resides. The command is synchronous, and the TCB waits for a response from CICS. Once a pipe is opened, any number of DPL requests can be issued before the pipe is closed. To the server program, the link request appears just like a standard EXEC CICS LINK request from another CICS region, and it is not aware that it is sent from a non-CICS client program using EXCI.

Syntax



DPL_Request command parameters

version_number

A full-word binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See page 192 for copybook details.

return_area

A 5-word output area to receive response and reason codes, and a message pointer field. For more details, see “Return area for the EXCI CALL interface” on page 192.

user_token

A 1-word input area specifying the user token returned to the client program on the Initialize_User command.

call_type

A 1-word input area indicating the function of the command. This must be set to 6 in the client program to indicate that the pipe is now being used for the DPL_Request call.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is DPL_REQUEST. See page 192 for copybook details.

pipe_token

A 1-word input area specifying the token returned by EXCI on the Allocate_Pipe command. It represents the pipe being used for the DPL_Request call.

pgmname

The 8-character name of the CICS application program being called as the server program.

This is either the name as specified on a predefined PROGRAM resource definition installed in the CICS server region, or as it is known to a user-written autoinstall program if the program is to be autoinstalled. The program can be defined in the CICS server region as a local program, or it can be defined as remote. Programs defined as remote enable “daisy-chaining”, where EXCI-CICS DPL calls become EXCI-CICS-CICS DPL calls.

COMMAREA
A variable length input area for the communications area (COMMAREA) between the client and server programs. The length is defined by *COMMAREA_len*.

This is the storage area that contains the data to be sent to the CICS application program. This area is also used to receive the updated COMMAREA from the CICS application program (the server program).

This parameter is optional. If it is not required, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see “Example of EXCI CALLs with null parameters” on page 193.

COMMAREA_len
A full-word binary input area. This parameter specifies the length of the COMMAREA. It is also the length of the server program’s COMMAREA (EIBCALEN).

If you specify a COMMAREA, you must also specify this parameter to define the length.

If you don't specify a COMMAREA, this parameter is ignored.

data_len
A full-word binary input area. This parameter specifies the length of contiguous storage, from the start of the COMMAREA, to be sent to the server program.

This parameter restricts the amount of data sent to the server program, and should be used to optimize performance if, for example, the COMMAREA is large but the amount of data being passed is small.

Note that on return from the server program, the EXCI data transformer program ensures that the COMMAREA in the non-CICS client program is the same as that of the server program. This caters for the following conditions:

- The data returned is **more** than the data passed in the original COMMAREA.
- The data returned is **less** than the data passed in the original COMMAREA.
- There is **no data** returned because it is unchanged.
- The server is returning **null data**.

The value of *data_len* must not be greater than the value of *COMMAREA_len*. A value of zero is valid and results in no data being sent to the server program.

If you don't specify a COMMAREA, this parameter is ignored.

transid

A 4-character input area containing the id of the CICS mirror transaction under which the server program is to run. This transaction must be defined to the CICS server region, and its definition **must observe the following rules**:

- It must *not* specify the server program as the initial program of the TRansaction
- It *must* specify the mirror program DFHMIRS, and the profile DFHCICSA.

Failure to specify DFHMIRS as the initial program means that a COMMAREA passed from the client application program is not passed to the CICS server program. Furthermore, the DPL request fails and the client application program receives a response of SYSTEM_ERROR and reason SERVER_PROTOCOL_ERROR.

+ The DFHCICSA profile specifies the correct value for the INBFMH parameter,
+ which must be specified as INBFMH(ALL) for a mirror transaction.

When the CICS server region receives a DPL request, it attaches the mirror transaction and invokes DFHMIRS. The mirror program then passes control to the requested server program, passing the COMMAREA supplied by the client program. The COMMAREA passed to the server program is primed with the data only, the remainder of the COMMAREA being set to nulls.

The purpose of the *transid* parameter is to distinguish between different invocations of the server program. This enables you to run different invocations of the server program under transactions that specify different attributes. For example, you can vary the transaction priorities, or the security requirements.

A transid is optional. By default, the CICS server region uses the CICS-supplied mirror transaction, CSMI. If you don't want to specify *transid*, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see “Example of EXCI CALLs with null parameters” on page 193.

uowid

An input area containing the APPC unit-of-work identifier.

This parameter is optional. If you don't want to specify *uowid*, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see “Example of EXCI CALLs with null parameters” on page 193.

If specified, the *uowid* parameter is passed to the CICS server region, which uses it as the APPC UOWID for the first unit of work executed by the CICS server program. If the server program issues intermediate syncpoints before returning to the client program, CICS uses the supplied *uowid* for the subsequent units of work, but with the two byte sequence number incremented for each new logical unit of work. If the CICS server program updates remote resources, the client-supplied APPC UOWID is distributed to the remote systems that own the resources.

The *uowid* parameter is supplied on the EXCI CALL interface for correlation purposes only, to allow units of work that originated from a particular client program to be identified in CICS. The *uowid* is not provided for recovery

purposes between CICS and the client program. No syncpoint coordination occurs between the client program and CICS, because all CICS server programs called from a client program run with SYNCONRETURN specified.

The *uowid* can be a maximum of 27 bytes long and has the following format:

- A 1-byte length field containing the overall length of the UOWID (excluding this field)
- A 1-byte length field containing the length of the logical unit name (excluding this field)
- A logical unit name field of variable length up to a maximum of 17 bytes
- The clock value—the middle 6 bytes of an 8-byte store clock (STCK) value
- A 2-byte sequence number.

If you omit a unit-of-work identifier (by specifying a null pointer) the external CICS interface generates one for you, consisting of the following:

- 1-byte length field set to X'1A'
- 1-byte LU length field set to X'11'
- A 17-byte LU name consisting of:
  - An 8-byte eye-catcher set to ‘DFHEXCIU’.
  - A 1-byte field contiaining a period (.)
  - A 4-byte field containing the MVSID, in characters, under which the client is running.
  - A 4-byte field containing the address space id (ASID) in which the MVS client program is running. The field contains the four-character EBCDIC representation of the two byte hex address space id.
- The clock value—the middle 6 bytes of an 8-byte store clock (STCK) value
- A two byte sequence number set to X'0001'.

userid

An 8-character input area containing the RACF userid for user security checking in the CICS region. The external CICS interface passes this userid to the CICS server region for user resource and command security checking in the server application program.

A userid is required only if the MRO connection specifies the ATTACHSEC(IDENTIFY) attribute. If the connection specifies ATTACHSEC(LOCAL), the CICS server region applies link security checking. See the *CICS/ESA CICS-RACF Security Guide* for information about link security on MRO connections.

This parameter is optional. However, if you don't specify a userid, the external CICS interface passes the security userid under which the client program is running. For example, if the client program is running as an MVS batch job, the external CICS interface obtains and passes the userid specified on the USER parameter of the JOB statement.

If you want to let *userid* default, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a

call that omits an optional parameter, see “Example of EXCI CALLs with null parameters” on page 193.

dpl_retarea
A 12-byte output area into which the DPL_Request processor places responses to the DPL request. Generally, these responses are from CICS, but in some cases the error detection occurs in the external CICS interface, which returns exception conditions that are the equivalent of those returned by an EXEC CICS LINK command.

This field is only meaningful in the following circumstances:

- The response field of the EXCI *return-area* has a zero value, or
- The EXCI *return-area* indicates that the server program has abended (response=USER_ERROR and reason=SERVER_ABENDED).

The 12-bytes form three fields, providing the following information:

1. The first field, a fullword value, contains a RESP value from the DPL_Request call.

   If the DPL_Request call reaches CICS, this field contains the EIBRESP value, otherwise it contains an equivalent response set by the external CICS interface. If this field is set by the external CICS interface, RESP is further qualified by a RESP2 value in the second field.

   A zero value is the normal response, which equates to EXEC_NORMAL in the return codes copybooks.

2. The second field, a fullword value, may contain a RESP2 value from the link request, further qualifying the RESP value in field 1.

   If the DPL_Request call reaches CICS, the RESP2 field is always null (CICS does not return RESP2 values across MRO links).

   If the RESP field is set by the external CICS interface, it is further qualified by a RESP2 value in this second field. For example, if the *data_len* parameter specifies a value greater than the *COMMAREA_len* parameter, the external CICS interface returns the value 22 (which equates to EXEC_LENGERR in the return codes copybooks), and a RESP2 value of 13.

   See the LINK conditions in *CICS/ESA Application Programming Reference* manual for details of the possible RESP and RESP2 values.

   Note: Special use of the RESP2 field by the data transformer program. If any error occurs in the transformer, the error is returned in RESP2.

3. The third field, a 4-character field, contains:

   - The abend code if the server program abended
   - 4 blanks if the server program did not abend.

If a server program abends, it is backed out to its last syncpoint which may be the start of the task, or an intermediate syncpoint. The server program can issue intermediate syncpoints because SYNCONRETURN is forced.

DPL_opts
A 1-byte input area indicating options to be used on the DPL_Request call.

For CICS/ESA 4.1, X'80' is the only valid option, and it indicates that SYNCONRETURN is specified. SYNCONRETURN is mandatory.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is SYNCONRETURN. See page 192 for copybook details.

SYNCONRETURN specifies that the server region is to take a syncpoint on successful completion of the server program. Note that although
+ SYNCONRETURN is mandatory, this does **not** prevent a server program from
+ taking its own explicit syncpoints.

Response and reason codes on the DPL call

For all non-zero response codes, a unique reason code value identifies the reason for the response.

The following is a summary of the response and reason codes that the external CICS interface can return on the DPL call:

Response OK:

0 NORMAL

Response WARNING:

6 IRP_IOAREA_FM_FAILURE
7 SERVER_TERMINATED

Repsonse RETRYABLE:

203 NO_CICS

Response USER_ERROR:

401 INVALID_CALL_TYPE
402 INVALID_VERSION_NUMBER
404 INVALID_USER_TOKEN
406 PIPE_NOT_OPEN
407 INVALID_USERID
408 INVALID_UOWID
409 INVALID_TRANSID
414 IRP_ABORT_RECEIVED
415 INVALID_CONNECTION_DEFN
416 INVALID_CICS_RELEASE
417 PIPE_MUST_CLOSE
418 INVALID_PIPE_TOKEN
421 RUNNING_UNDER_AN_IRB
422 SERVER_ABENDED

Response SYSTEM_ERROR:

612 TRANSFORM_1_ERROR
613 TRANSFORM_4_ERROR
614 IRP_NULL_DATA_RECEIVED
615 IRP_NEGATIVE_RESPONSE
616 IRP_SWITCH_PULL_FAILURE
617 IRP_IOAREA_GM_FAILURE
619 IRP_BAD_IOAREA
620 IRP_PROTOCOL_ERROR
622 ESTAE_SETUP_FAILURE

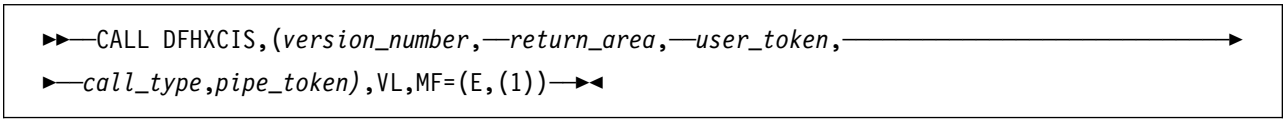
623 ESTAE_INVOKED
624 SERVER_TIMEDOUT
625 STIMER_SETUP_FAILURE
626 STIMER_CANCEL_FAILURE
629 SERVER_PROTOCOL_ERROR

CLOSE_PIPE

Function

Disconnect an open pipe from CICS. The pipe remains in an allocated state, and its tokens remain valid. To reuse a closed pipe, you must first reissue an Open_Pipe command using the pipe token returned on the Allocate_Pipe command for the pipe. Pipes should not be left open when not in use because this prevents CICS from shutting down its IRC facility in an orderly manner. Therefore, the Close_Pipe command should be issued as soon as possible after all DPL_Request calls have completed.

Syntax



Close_Pipe command parameters

version_number

A full-word binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See page 192 for copybook details.

return_area

A 5-word output area to receive response and reason codes, and a message pointer field. For more details, see “Return area for the EXCI CALL interface” on page 192.

user_token

The 1-word input area specifying the token, returned to the client program by EXCI on the Initialize_User command, that represents the user of the pipe being closed.

call_type

A 1-word input area indicating the function of the command. This must be set to 4 in the client program to indicate that this is a Close_Pipe command.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is CLOSE_PIPE. See page 192 for copybook details.

pipe_token

A 1-word input area specifying the token, returned to the client program by EXCI on the original Allocate_Pipe command, that represents the pipe being closed.

Response and reason codes on the Close_Pipe call

For all non-zero response codes, a unique reason code value identifies the reason for the response.

The following is a summary of the response and reason codes that the external CICS interface can return on the Close_Pipe call:

Response OK:

0 NORMAL

Response WARNING

2 PIPE_ALREADY_CLOSED

Response USER_ERROR

401 INVALID_CALL_TYPE
402 INVALID_VERSION_NUMBER
404 INVALID_USER_TOKEN
418 INVALID_PIPE_TOKEN
421 RUNNING_UNDER_AN_IRB

Response SYSTEM_ERROR

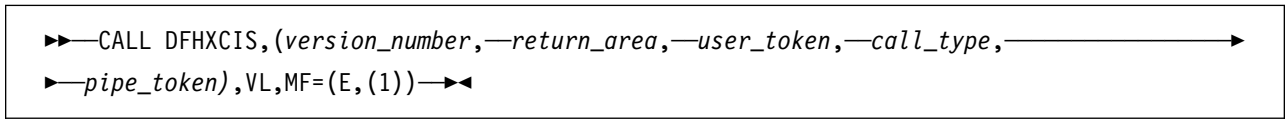
610 IRC_DISCONNECT_FAILURE
622 ESTAE_SETUP_FAILURE
623 ESTAE_INVOKED

DEALLOCATE_PIPE

Function

Deallocate a pipe from CICS. On completion of this command, the pipe can no longer be used, and its associated tokens are invalid. This command should be issued for pipes that are no longer required. This command frees storage associated with the pipe.

Syntax



Deallocate_Pipe command parameters

version_number

A full-word binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See page 192 for copybook details.

return_area

A 5-word output area to receive response and reason codes, and a message pointer field. For more details, see “Return area for the EXCI CALL interface” on page 192.

user_token

A 1-word input area containing the token returned on the Initialize_User command.

call_type

A 1-word input area indicating the function of the command. This must be set to 5 in the client program to indicate that this is a Deallocate_Pipe command.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is DEALLOCATE_PIPE. See page 192 for copybook details.

pipe_token

A 1-word input area containing the token passed back on the original Allocate_Pipe command, that represents the pipe now being deallocated.

Response and reason codes on the Deallocate_Pipe call

For all non-zero response codes, a unique reason code value identifies the reason for the response.

The following is a summary of the response and reason codes that the external CICS interface can return on the Deallocate_Pipe call:

Response OK:

0 NORMAL

Response WARNING:

5 XCPIPE_FREEMAIN_ERROR
6 IRP_IOAREA_FM_FAILURE

Response USER_ERROR:

401 INVALID_CALL_TYPE
402 INVALID_VERSION_NUMBER
404 INVALID_USER_TOKEN
405 PIPE_NOT_CLOSED
418 INVALID_PIPE_TOKEN
421 RUNNING_UNDER_AN_IRB

Response SYSTEM_ERROR:

611 IRC_LOGOFF_FAILURE
622 ESTAE_SETUP_FAILURE
623 ESTAE_INVOKED

Response code values

The values that can be returned in the response field are shown in Table 11 (all values are specified in decimal):

Table 11. EXCI response codes (returned in response field of return_area)

Value	Meaning	Explanation
0	OK	For all EXCI CALL commands other than the DPL command, the call was successful. If an OK response is received for a DPL command, you must also check *Dpl_retarea* to ensure CICS did not return a condition code. If the EIBRESP field of *Dpl_retarea* is zero, the DPL call was successful.
4	WARNING	The external CICS interface detected an error, but this did not stop the CALL command completing successfully. The reason code field describes the error detected.
8	RETRYABLE	The EXCI CALL command failed. This class of failure relates to errors in the setup of the system environment, and not errors in the external CICS interface or client program. The reason code documents the specific error in the environment setup. The external CICS interface command can be reissued without changing the client program once the environment error has been corrected. The environmental errors concerned are ones that do not require an MVS re-IPL. Each reason code value for a RETRYABLE response documents whether the CALL can be reissued directly, or whether the pipe being used has to be closed and reopened first.
12	USER_ERROR	The EXCI CALL command failed. This class of error means there is an error either in the client program, or in the CICS server program, or in the CICS server region. An example of an error in the CICS server system would be a failed security check, or an abend of the CICS server program, in which case the abend code is set in the abend code field of *DPL_retarea*. Each reason code value for a response of USER_ERROR explains whether the command can be reissued directly, or whether the pipe being used has to be closed and reopened first.
16	SYSTEM_ ERROR	The EXCI CALL command failed. This class of error means that the external CICS interface has detected an error. The reason code value identifies the specific error. If the error can be corrected, then the command can be reissued. Each reason code value for a SYSTEM_ERROR response explains whether the command reissued directly, or whether the pipe being used has to be closed and reopened first.

Return area for the EXCI CALL interface

The format of the 5-word return area for the EXCI CALL interface is as follows:

1. One-word response field
2. One-word reason field
3. Two one-word subreason fields—subreason field-1 and subreason field-2
4. One-word CICS message pointer field. This is zero if there is no message present. If a message is present, this field contains the address of the storage area containing the message, which is formatted as follows:
   - A 2-byte LL field. LL is the length of the message plus the length of the LLBB field.
   - A 2-byte BB field, set to binary zero.
   - A variable length field containing the text of the message.

Return area and function call EQUATE copybooks

CICS provides four language-specific copybooks that map the storage areas for the *return_area* and *dpl_retarea* parameters of the EXCI CALL commands. The copybooks also provide EQUATE statements for each type of EXCI CALL.

These copybooks, and the libraries they are supplied in for the supported languages, are shown in Table 12.

Table 12. Supplied copybooks of return areas and equated names

Copybook name	Language	Library
DFHXCPLD	Assembler	CICS410.SDFHMAC
DFHXCPLH	C	CICS410.SDFHC370
DFHXCPLO	COBOL	CICS410.SDFHCOB
DFHXCPLL	PL/I	CICS410.SDFHPL1

Return codes

All the possible return codes are contained in a CICS-supplied copybook, which you must include in the program source of your external, non-CICS program. The names of the copybooks for the supported languages, and the libraries they are supplied in, are shown in Table 13.

Table 13. Supplied copybooks of RESPONSE and REASON codes

Copybook name	Language	Library
DFHXCRCD	Assembler	CICS410.SDFHMAC
DFHXCRCH	C	CICS410.SDFHC370
DFHXCRCO	COBOL	CICS410.SDFHCOB
DFHXCRCL	PL/I	CICS410.SDFHPL1

Dpl_Retarea return codes

These are the same as for CICS-to-CICS EXEC CICS DPL commands but with the following additions for the EXCI call interface:

Table 14. Exceptional conditions. RESP and RESP2 values returned to DPL_RETAREA

Condition	RESP2	Meaning
INVREQ	21	SYNCONRETURN_NOT_SPECIFIED
LENGERR	22	COMMAREA_LEN_TOO_BIG
LENGERR	23	COMMAREA_BUT_NO_COMMAREA_LEN

SYSIDERR also may be returned on an EXCI DPL request if the DPL request was to a program defined as remote and the link between CICS systems is down. In this situation, SYSIDERR is returned in the first word of the DPL_Retarea (code 53), and the reason for the error, as documented in the *CICS/ESA Application Programming Reference* manual in the SYSIDERR section of the notes on EIBRCODE, is placed in the second word of this area.

TERMERR also may be returned on an EXCI DPL request if the DPL request was to a program defined as remote, and an unrecoverable error occurs during conversation with the mirror on the remote CICS system. For example, suppose client program BATCH1 issues an EXCI DPL request to CICSA for program PROG1, which is defined as remote, and the request is function-shipped to CICSB where the program resides. If the session between CICSA and CICSB fails, or CICSB itself fails whilst executing the program PROG1, then TERMERR is returned to CICSA, and in turn to BATCH1.

No unique EXCI_DPL_RESP2 values are returned for TERMERR, PGMIDERR, NOTAUTH, and ROLLBACK.

Example of EXCI CALLs with null parameters

This section describes an example of an EXCI DPL call with an optional parameter omitted. It is a COBOL example, where *userid* and *uowid* are omitted, and a null pointer is passed in place of the missing parameters.

DPL CALL without userid and uowid (COBOL): In this example, the DPL parameters used on the call are defined in the WORKING-STORAGE SECTION, as follows.

```
DPL parameter   COBOL variable
version_number  01 VERSION-1          PIC S9(8) COMP VALUE 1.
return_area     01 RETAREA.           structure
user_token      01 USER-TOKEN         PIC S9(8) COMP VALUE ZERO.
call_type       03 DPL-REQUEST        PIC S9(8) COMP VALUE 6.
pipe_token      01 PIPE-TOKEN         PIC S9(8) COMP VALUE ZERO.

pgmname         01 TARGET-PROGRAM     PIC X(8) VALUE "DFH$AXCS".
commarea        01 COMMAREA.          structure
commarea_len    01 COMM-LENGTH        PIC S9(8) COMP VALUE 98.
data_len        01 DATA-LENGTH        PIC S9(8) COMP VALUE 18.
transid         01 TARGET-TRANSID     PIC X(4) VALUE "EXCI".

dpl_retarea     01 DPL-RETAREA.       structure
dpl_opts        01 SYNCONRETURN       PIC X VALUE X"80".
```

The variable used for the null address is defined in LINKAGE SECTION, as follows:

```
LINKAGE SECTION.
 01  NULL-PTR        USAGE IS POINTER.
```

Using the data names specified in WORKING-STORAGE SECTION as described above, and the NULL-PTR name as described in the LINKAGE SECTION, the following invocation of the DPL function omits the *uowid* and the *userid* parameters, and replaces them in the parameter list with the NULL-PTR variable:

```
 DPL-SECTION.
     SET ADDRESS OF NULL-PTR TO NULLS.
*
     CALL 'DFHXCIS' USING  VERSION-1    RETAREA      USER-TOKEN
                           DPL-REQUEST    PIPE-TOKEN   TARGET-PROGRAM
                           COMMAREA       COMM-LENGTH  DATA-LENGTH
                           TARGET-TRANSID NULL-PTR     NULL-PTR
                           DPL-RETAREA    SYNCONRETURN.
```

This example is taken from the CICS-supplied sample external CICS interface program, DFH0CXCC, which is supplied in CICS410.SDFSAMP. For an example of how to omit the same parameters from the DPL call in the other supported languages, see the following sample programs:

DFH$AXCC The assembler sample
DFH$PXCC The PL/I sample
DFH$DXCC The C sample

Changes to the CICS application programming interface (API)

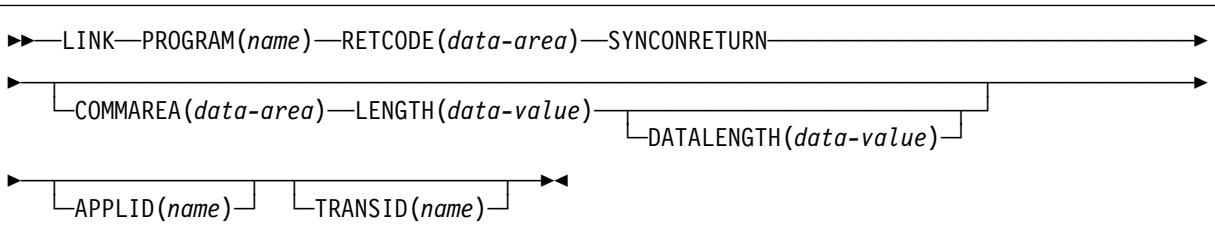
The external CICS interface provides a single, composite command that performs all six commands of the EXCI CALL interface in one invocation. It is simpler to code for a single DPL request to a CICS server region. This command takes the same form as the distributed program link command of the CICS command-level application programming interface.

EXEC CICS LINK command

Function

Link from an MVS client program to the specified server program in a CICS region.

Syntax



Conditions:
INVREQ, LENGERR, LINKERR, NOTAUTH, PGMIDERR, ROLLEDBACK, SYSIDERR, WARNING

LINK PROGRAM options

With the exception of the APPLID and RETCODE parameters, the external CICS interface parameters for an EXEC CICS LINK command are the same as for a CICS-CICS DPL command. For information about the following, see the *CICS/ESA Application Programming Reference* manual:

- PROGRAM
- SYNCONRETURN
- COMMAREA
- LENGTH
- DATALENGTH
- TRANSID

Note that the LENGTH and DATALENGTH parameters specify half-word binary values, unlike the corresponding *COMMAREA_len* and *data_len* parameters of the EXCI CALL interface, which specify full-word values.

An external CICS interface EXEC CICS LINK command always uses a generic connection.

The parameters that are unique to the external CICS interface form of the LINK command, or where the meaning varies from that of the CICS–CICS DPL command, are as follows:

APPLID
specifies the generic APPLID of the target CICS server region.

Although an applid is required for an external CICS interface command, this parameter is optional on the LINK command itself because you can also specify it in the user-replaceable module, DFHXCURM. If you omit the generic APPLID from the LINK command, you must ensure it is specified by the user-replaceable module, DFHXCURM, on the URMAPPL parameter. You can also use the URMAPPL parameter in DFHXCURM to override an applid specified on the LINK command. See "The external CICS interface user-replaceable module, DFHXCURM" on page 202 for information about the URMAPPL parameter.

RETCODE
specifies a 20-byte area into which the external CICS interface places return code information. This area is formatted into five 1-word fields as follows:

RESP The primary response code indicating whether the external CICS interface LINK command caused an exception condition during its execution.
RESP2 The secondary response code that further qualifies, where necessary, some of the conditions raised in the RESP parameter.
ABCODE Contains a valid CICS abend code if the server program abended in the server region.
MSGLEN Indicates the length of the message (if any) issued by the CICS server region during the execution of the server program. Note that the length is the actual length of the message text only, and does not include this one-word length field.
MSGPTR This is the address of the message text returned by the CICS server region.

Note: MSGLEN and MSGPTR are only valid on a LINKERR condition, and for the RESP2 value 414.

SYNCONRETURN
specifies that the CICS server region, named on the APPLID parameter, is to take a syncpoint on successful completion of the server program.

SYNCONRETURN is mandatory for an external CICS interface LINK command.

Exception conditions

Most of the exception conditions that are returned on the external CICS interface LINK command are the same as for the CICS-to-CICS distributed program link command. The exception conditions that are specific to the external CICS interface are as follows:

- The conditions WARNING and LINKERR are specific to the external CICS interface
- Some of the RESP2 values on the error conditions INVREQ and LENGERR are specific to the external CICS interface

Table 15 lists all the exception conditions and RESP2 values that are specific to the external CICS interface.

Table 15 (Page 1 of 4). Exceptional conditions. RESP and RESP2 values returned from the EXEC API

Condition (RESP)	RESP2	Meaning
INVREQ	21	SYNCONRETURN has not been specified
LENGERR	22	COMMAREA length greater than 32763 bytes specified
	22	COMMAREA specified but no length COMMAREA length specified
WARNING	401	Invalid *call_type* parameter value specified on Close_Pipe or Deallocate_Pipe call
	402	Invalid *version_number* parameter specified on Close_Pipe or Deallocate_Pipe call
	404	Invalid *user_token* specified on Close_Pipe or Deallocate_Pipe call
	405	A Deallocate_Pipe call has been issued against a pipe that is not yet closed
	418	An invalid pipe token has been issued on a Close_Pipe or Deallocate_Pipe call
	421	A Close_Pipe or Deallocate_Pipe command has been issued under an IRB
	610	There has been a CICS IRP logoff failure on a Deallocate_Pipe call
	611	There has been a CICS IRC disconnect failure on a Close_Pipe call
	622	There has been an MVS ESTAE setup failure on a Close_Pipe or Deallocate_Pipe call

Table 15 (Page 2 of 4). Exceptional conditions. RESP and RESP2 values returned from the EXEC API

Condition (RESP)	RESP2	Meaning
WARNING (*cont'd*)	623	A program check on a Close_Pipe or Deallocate_Pipe call has caused the ESTAE to be invoked
LINKERR	201	Command has been issued on an MVS image which has had no IRC activity since the previous IPL
	202	There are no available sessions
	203	CICS has not yet been brought up, or has not yet opened IRC
	401	Invalid parameter
	402	Invalid version number
	403	User name is all blanks
	404	Invalid address in user token
	405	Command has been issued against a pipe that is not closed
	406	Command has been issued against a pipe that is not open
	407	Userid of all blanks has been passed
	408	Error in UOWID parameter
	409	Transid consisting of all blanks or zero has been passed
	410	Load of message module, DFHMEBM, failed
	411	Load of message module, DFHMET4E, failed
	412	Load of DFHXCURM failed
	413	Load of DFHXCTRA failed
	414	If run as a CICS-to-CICS linked-to program, this server program would have resulted in an error with a appropriate message sent to the terminal. Running the program as an EXCI server program returns the message addressed by the MSGPTR field of the RETCODE area
	415	Target connection is an MRO connection, not an EXCI connection
	416	Command has been issued against a pre-CICS/ESA 4.1 system
	417	Command has been issued against a pipe in the MUST CLOSE state. Further EXCI EXEC CICS LINK commands will have unpredictable results and are, therefore, not permitted
	418	Pipe_token does not address an XCPIPE control block, or there is a mismatch between user_token and pipe_token
	419	CICS runs, or did run, under the TCB that this command is attempting to use. This is not permitted and the command fails
	420	Load of DFHXCOPT failed
	421	The command has been issued under an MVS IRB, which is not permitted
	422	The server has abended

Table 15 (Page 3 of 4). Exceptional conditions. RESP and RESP2 values returned from the EXEC API

Condition (RESP)	RESP2	Meaning
LINKERR (*cont'd*)	601	A GETMAIN of working storage failed. This error leads to user abend 408
	602	A GETMAIN failed. This error leads to user abend 403
	603	A GETMAIN failed. This error leads to user abend 410
	604	A GETMAIN failed
	605	A GETMAIN for the VERIFY block failed. This error leads to user abend 409
	606	An SSI verify request (to obtain CICS SVC instruction) failed. This error leads to user abend 405
	607	An SVC call failed. This error leads to user abend 406
	608	Logon to IRP failed
	609	Connect to IRP failed
	610	Disconnect from IRP failed
	611	Logoff from IRP failed
	612	Invalid data input to transformer_1
	613	Invalid data input to transformer_4
	614	CICS has responded but has not sent any data
	615	CICS cannot satisfy the request
	616	IRP_SWITCH_PULL request (to read data sent from CICS into a larger input/output area) has failed
	617	A GETMAIN for a larger input/output area failed
	619	IRP has had a problem with the input/output area passed from the client program
	620	IRP has disconnected from EXCI
	621	A DISCONNECT command is issued in an error situation following an IRP CONNECT. The DISCONNECT has failed, indicating a serious error
	622	XCPRH ESTAE set-up command failed This error leads to user abend 402.
	623	XCPRH ESTAE invoked due to program check during the processing of this command. ESTAE attempts backout and takes a SYSMDUMP. Further requests are permitted although the pipe is now in a MUST CLOSE state
	624	The DPL request has been passed to CICS but the time specified in DFHXCOPT has been exceeded. The request is aborted
	625	An MVS STIMERM macro call failed
	626	An MVS STIMERM CANCEL request failed
	627	The CICS SVC is at the incorrect level This error leads to user abend 407
	628	DFHIRP is at the incorrect level
	903	AN XCEIP ESTAE set-up command failed

Table 15 (Page 4 of 4). Exceptional conditions. RESP and RESP2 values returned from the EXEC API

Condition (RESP)	RESP2	Meaning
LINKERR (*cont'd*)	904	The server program abended with the abend code in the ABCODE field of the RETCODE area
	905	An XCEIP ESTAE invoked

Translation required for EXEC CICS LINK command

Application programs that use the EXEC CICS LINK form of the external CICS interface command must translate their programs before assembly or compilation. You do this using the version of the CICS translator that is appropriate for the language of your client program, specifying the translator option EXCI.

The translator option EXCI is mutually exclusive with the CICS and DLI options.

For more information about translating programs that contain EXEC CICS commands, see the *CICS/ESA Application Programming Guide*.

For information about compiling and link-editing external CICS interface client programs, see page 209.

End of General-use programming interface

Changes to resource definition interfaces

There are some changes to CICS resource definition to support the external CICS interface. These are:

- A new attribute, CONNTYPE, is added to the CONNECTION resource definition
- A new value, EXCI, is added to the PROTOCOL attribute of the CONNECTION and SESSIONS resource definitions.

CONNECTION resource definition changes

A new value, EXCI, is introduced on the PROTOCOL attribute of the CONNECTION resource definition. This indicates that the connection is intended for use by a program using the external CICS interface.

A new attribute, CONNTYPE, is also introduced on the CONNECTION resource definition. For EXCI connections, this indicates whether the connection is generic or specific. It is not to be used for any protocol other than the external CICS interface.

```
 Connection   ==> ....
 Group        ==> ........
 DEscription  ==> ................................................

CONNECTION IDENTIFIERS
 Netname      ==> ........
 INDsys       ==> ....

REMOTE ATTRIBUTES
  ...

CONNECTION PROPERTIES
 ACcessmethod ==> IRC              Vtam | IRc | INdirect | Xm
 Protocol     ==> EXCI             Appc | Lu61 | EXCI
 Conntype     ==>                  Generic | Specific
 SInglesess   ==> No               No | Yes
 ...
```

Figure 12. The DEFINE panel for CONNECTION

CONNTYPE({SPECIFIC|GENERIC})
For external CICS interface connections, indicates the nature of the connection.

SPECIFIC
The connection is for communication from a non-CICS client program to the CICS region, and is specific. A specific connection is an MRO link with one or more sessions dedicated to a single user in a client program. For a specific connection, NETNAME is mandatory.

GENERIC
The connection is for communication from a non-CICS client program to the CICS system, and is generic. A generic connection is an MRO link with a number of sessions to be shared by multiple EXCI users. For a generic connection you cannot specify the NETNAME attribute.

Note: You must install only one generic EXCI connection in a CICS region.

PROTOCOL({APPC|LU61|EXCI|blank})
The type of protocol that is to be used for the link.

blank
For MRO between CICS regions. You must leave the PROTOCOL blank for MRO, and on the SESSIONS definition you must specify LU6.1 as the PROTOCOL.

APPC (LUTYPE6.2 protocol)
Advanced program-to-program communication, or APPC protocol. This is the default value for ACCESSMETHOD(VTAM). Specify this for CICS-CICS ISC.

LU61
LUTYPE6.1 protocol. Specify this for CICS-CICS ISC or CICS-IMS ISC, but not for MRO.

EXCI
The external CICS interface. Specify this to indicate that this connection is for use by a non-CICS client program using the external CICS interface.

SESSIONS resource definition changes

A new value, EXCI, is introduced on the PROTOCOL attribute of the SESSIONS resource definition. This indicates that the sessions on the named connection are intended for use by a program using the external CICS interface.

```
 Sessions     ==> ........
 Group        ==> ........
 DEscription  ==> ............................................

SESSION IDENTIFIERS
 Connection   ==> ....
 SESSName     ==> ....
 NETnameq     ==> ........
 MOdename     ==> ........

SESSION PROPERTIES
 Protocol     ==> Appc              Appc | Lu61 | EXCI
  ...
```

Figure 13. The DEFINE panel for SESSIONS

PROTOCOL({APPC|LU61|EXCI})
Indicates the type of protocol that is to be used for an intercommunication link (ISC or MRO).

APPC (LUTYPE6.2)
Advanced program-to-program communication (APPC) protocol. Specify this for CICS-CICS ISC.

LU61
LUTYPE6.1 protocol. Specify this for CICS-CICS ISC, for CICS-IMS, or for MRO.

EXCI
The external CICS interface. Specify this to indicate that the sessions are for use by a non-CICS client program using the external CICS interface. If you specify EXCI, you must leave SENDCOUNT blank.

SENDCOUNT(blank|number)
The number of MRO or LUTYPE6.1 sessions that usually send before receiving.

For MRO, send sessions must send before they can receive.

blank
These sessions can receive only; there are no send sessions.

You must leave this field blank when the sessions are on an external CICS interface (EXCI) connection.

number
Specifies the number of send sessions on connections that specify blank or LU61 on the protocol parameter of the CONNECTION definition. CICS

uses the number to generate the last two or three characters of the session names (see SENDPFX for details).

If you are using the default send prefix (>), or your own 1-character prefix, specify a number in the range 1 through 999.

If you specify a 2-character prefix, the number is restricted to the range 1 through 99.

Except for external CICS interface (EXCI) connections the SENDCOUNT in this system should equal RECEIVECOUNT in the other system.

Changes to user-replaceable modules

Product-sensitive programming interface

There are no changes to existing CICS user-replaceable modules. However, the external CICS interface introduces a new user-replaceable module, DFHXCURM, for use in the client program address space. The purpose of the user-replaceable module is to solve problems of availability.

The external CICS interface user-replaceable module, DFHXCURM

DFHXCURM is invoked in the non-CICS region during the processing of Allocate_Pipe commands, and after the occurrence of any re-tryable error. The re-tryable responses are:

- The target CICS region is not available
- There are no pipes available on the target CICS region
- There has been no IRC activity since the MVS IPL

As supplied, DFHXCURM is effectively a dummy program because of a branch instruction that bypasses the sample logic and returns control to the external CICS interface caller. To use the sample logic, remove the branch instruction and assemble and link-edit the module. Customizing DFHXCURM allows you to do the following:

- When invoked during Allocate_Pipe processing, you can change the specified CICS APPLID, in order to route the request to another CICS system.
- When invoked after a re-tryable error you can store information regarding CICS availability. You can then use this information on the next invocation of DFHXCURM for Allocate_Pipe processing, so that you can decide to which CICS system to route the request.

DFHXCURM is called using standard MVS register conventions, with register 1 containing the address of the parameter list, and register 14 the return address of the caller. The parameters addressed by register 1 are mapped in the EXCI_URM_PARMS DSECT, which is contained within the DFHXCPLD copybook. The parameters passed to DFHXCURM are as follows:

URMINV
The address of a full-word that contains the reason for the invocation of DFHXCURM, defined by the following equates:

```
URM_ALLOCATE     EQU 1  This invocation is for an Allocate_Pipe
URM_NO_CICS      EQU 2  The target CICS region is not available
URM_NO_PIPE      EQU 3  There are no pipes available
URM_NO_CICS_IRC  EQU 4  There has been no IRC activity since the MVS IPL
```

URMCICS
The address of an 8-byte area that contains the generic APPLID of the target CICS system, as specified on the *CICS_applid* parameter of the Allocate_Pipe command, or on the APPLID parameter of the EXEC CICS LINK command.

When specified by one of these commands, you can change the APPLID to that of a different target CICS region.

If the *CICS_applid* parameter is omitted from the Allocate_Pipe call, or APPLID is omitted from the EXEC CICS LINK command, the field addressed by this parameter contains 8 blanks. In this case, you must specify an APPLID in DFHXCURM before returning control to the caller.

URMAPPL
The address of an 8-byte area that contains the client program's user name as specified on the *my_name* parameter of the Initialize_User command. Note that if DFHXCURM is invoked for an EXEC CICS LINK command, this name is always set to DFHXCEIP.

URMPROG
The address of an 8-byte area that contains the name of the target program (if available). This name is available only if DFHXCURM is invoked for an EXEC CICS LINK command. For an external CICS interface Allocate_Pipe command, the program name is not known until the DPL call is issued.

URMOPTS
The address of a 1-byte area that contains the allocate options, which can be X'00' or X'80', as specified on the *allocate_opts* parameter. This address is valid for an Allocate_Pipe request only.

URMANCH
The address of a 4-byte area that is provided for use by DFHXCURM only. A typical use for this is to store a global anchor address of an area used to save information across a number of invocations of DFHXCURM. For example, you can GETMAIN the necessary storage and save the address in the 4-byte area addressed by this parameter. The initial value of the 4-byte area is set to zero.

Loss of abend handling by ESTAE

You are recommended not to use the MVS XCTL macro to pass control from your customized DFHXCURM, because this causes the deletion of the ESTAE environment from the client program address space.

End of Product-sensitive programming interface

External CICS interface options table, DFHXCOPT

A new CICS table, generated by the DFHXCOPT macro, enables you to specify a number of parameters that are required by the external CICS interface.

CICS provides a sample DFHXCOPT table, which you can modify to suit your own requirements. You assemble and link-edit the modified DFHXCOPT table into a suitable library in the STEPLIB concatenation of the job that runs the MVS client program. Unlike the tables you specify for your CICS regions, the DFHXCOPT table cannot be suffixed, and the external CICS interface component loads the first table of this name that it finds in the STEPLIB concatenation.

Table 16 shows the format of the DFHXCOPT macro and its parameters.

Table 16. The DFHXCOPT macro parameters

	DFHXCO	TYPE={**CSECT**|DSECT} [,CICSSVC={**0**|number}] + [,CONFDATA={**SHOW**|HIDETC}] [,DURETRY={**30**|number-of-seconds}] [,GTF={**OFF**|ON}] [,MSGCASE={**MIXED**|UPPER}] [,TIMEOUT={**0**|number}] [,TRACE={**OFF**|1|2}] [,TRACESZE={**16**|number-of-kilobytes}] [,TRAP={**OFF**|ON}] You must terminate the parameters with the following END statement.
	END	DFHXCOPT

TYPE={CSECT|DSECT}

Indicates the type of table to be generated.

CSECT

A regular control section that is normally used.

DSECT

A dummy control section.

CICSSVC={0|number}

Specifies the CICS type 3 SVC number being used for MRO communication.

The external CICS interface must use the same SVC number that is in use by the CICS MRO regions that reside in the MVS image in which the client program is running.

If you do not specify a specific CICS SVC number, the external CICS interface determines the SVC in use for MRO by means of an MVS VERIFY command.

0 Specify zero to indicate that the external CICS interface is to obtain the CICS SVC number from MVS. This is the default.

You should only specify 0 when you are sure that at least one CICS region has logged on to DFHIRP during the life of the MVS IPL.

number
Specify the CICS SVC number, in the range 200—255, that is in use for CICS interregion communications. This must be the SVC number that is installed in the MVS image in which the client program is running (the local MVS).

If no MRO CICS regions have ever logged on to DFHIRP in the local MVS during the life of the IPL, you must specify the SVC number. If you allow this parameter to default, and the external CICS interface requests the SVC from MVS, the request will fail if no CICS region has logged on to DFHIRP.

This parameter is required in those MVS images that do not run any CICS regions, and the client program is issuing DPL requests to a server CICS region that resides in another MVS. In these circumstances, the client program logs on to the local DFHIRP using the locally defined SVC, and communicates with the remote CICS region using XCF/MRO.

Note: All CICS regions using MRO within the same MVS image must use the highest level of both DFHIRP and the CICS SVC, DFHCSVC. If your MRO CICSplex consists of CICS regions at different release levels, the DFHIRP and DFHCSVC installed in the LPA must be from highest release level of CICS within the CICSplex.

MVS client programs using the external CICS interface can communicate only with server regions running under CICS/ESA 4.1 or later.

+ **CONFDATA={SHOW|HIDETC}**
+ Code this parameter to indicate whether the external CICS interface is to
+ suppress (hide) user data that might otherwise appear in EXCI trace entries
+ output to GTF or in EXCI dumps. This option applies to the tracing of the
+ COMMAREA flowing between the EXCI client program and the CICS server
+ program.

+ **SHOW**
+ Data suppression is not in effect. User data is traced.

+ **HIDETC**
+ EXCI 'hides' user COMMAREA data from trace entries. Instead of the
+ COMMAREA data, the trace entry contains a character string stating that
+ the data has been suppressed.

DURETRY={30|number-of-seconds|0}
Specifies the total time, in seconds, that the external CICS interface is to continue trying to obtain an MVS system dump using the SDUMP macro.

DURETRY allows you to control whether, and for how long, the external CICS interface is to reissue the SDUMP if another address space in the same MVS system is already taking an SDUMP when the external CICS interface issues an SDUMP request.

In the event of an SDUMP failure, the external CICS interface reacts as follows:

- If MVS is already taking an SDUMP for another address space, and the DURETRY parameter is nonzero, the external CICS interface issues an MVS STIMERM macro to wait for five seconds, before retrying the SDUMP macro. The external CICS interface issues a message to say that it will retry the SDUMP every five seconds until the DURETRY time limit.

- If the SDUMP fails for any other reason such as:
  - There are no SYS1.DUMP data sets available, or
  - There are I/O errors preventing completion of the dump, or
  - The DURETRY limit expires while retrying SDUMP

  the external CICS interface issues a message to inform you that the SDUMP has failed, giving the reason why.

30 30 seconds allows the external CICS interface to retry up to six times (once every five seconds).

number-of-seconds
Code the total number of seconds (up to 32767 seconds) during which you want the external CICS interface to continue retrying the SDUMP macro. The external CICS interface retries the SDUMP, once every five seconds, until successful or until retries have been made over a period equal to or greater than the DURETRY value.

0 Code a zero value if you do not want CICS to retry the SDUMP.

GTF={OFF|ON}
Specifies whether all trace entries, normally written to the external CICS interface internal trace table, are also to be written to an MVS generalized trace facility (GTF) data set (if GTF tracing is active).

OFF
Trace entries are not to be written to GTF.

ON
Trace entries are to be written to GTF.

MSGCASE={MIXED|UPPER}
Specifies whether the DFHEXxxxx messages are to be issued in mixed or upper case.

MIXED
Code this if messages are to be issued in mixed case.

UPPER
Code this if messages are to be issued in upper case.

TIMEOUT={0|number}
Specifies the time interval, in hundredths of a second, during which the external CICS interface waits for a DPL command to complete.

0 Specifies that you do not want any time limit applied, and that the external CICS interface is to wait indefinitely for a DPL command to complete.

number
Specifies the time interval, in hundredths of a second, that the external CICS interface is to wait for a DPL command to complete. The number represents hundredths of a second, from 1 up to a maximum of 2 147 483 647. For example:

6000 Represents a timeout value of one minute

30000 Represents a timeout value of five minutes

60000 Represents a timeout value of ten minutes

TRACE={OFF|1|2}
Specifies whether you want external CICS interface internal tracing, and at what level.

OFF
External CICS interface internal tracing is not required. However, even with normal tracing switched off, exception trace entries are always written to the internal trace table.

1 Exception and level-1 trace entries are written to the internal trace table.

2 Exception, level-1, and level-2 trace entries are written to the internal trace table.

TRACESZE={16|number-of-kilobytes}
Specifies the size in kilobytes of the internal trace table for use by the external CICS interface. This table is allocated in virtual storage above 16MB. You should ensure that there is enough virtual storage for the trace table by specifying a large enough region size on the MVS REGION parameter.

16 16KB is the default size of the trace table, and also the minimum size.

number-of-kilobytes
The number of kilobytes of storage to be allocated for the internal trace table, in the range 16KB through 1 048 576KB. Subpool 1 is used for the trace table storage, which exists for the duration of the jobstep TCB. The table is page-aligned and occupies a whole number of pages. If the value specified is not a multiple of the page size (4KB), it is rounded up to the next multiple of 4KB.

TRAP={OFF|ON}
Specifies whether the service trap module, DFHXCTRA, is to be used. DFHXCTRA is supplied as a user-replaceable module, in which IBM service personnel can add code to trap errors.

OFF
Code this if you do not want to use DFHXCTRA.

ON
Code this if you require DFHXCTRA.

Problem determination

The following CICS messages are introduced to support the external CICS interface:

```
DFHIR3799

DFHEX0001         DFHEX0011
DFHEX0002         DFHEX0012
DFHEX0003         DFHEX0013
DFHEX0004         DFHEX0014
DFHEX0005         DFHEX0015
DFHEX0010         DFHEX0016
```

Messages DFH5502W and DFH5503E are extended to include the external CICS interface facility.

This facility introduces new translator messages, DFH7004I and DFH7005I.

For full details of these, see the *CICS/ESA Migration Guide*.

The external CICS interface outputs trace to two destinations: an internal trace table and an external MVS GTF data set. The internal trace table resides in the MVS address space of the client program. Trace data is formatted and included in any dumps produced by the external CICS interface.

New trace entries are written by the external trace interface to the internal trace table (if you have trace switched on). You can also choose to have the trace entries written to the MVS GTF data set. All EXCI trace entries are described in the *CICS/ESA External CICS Interface*.

The external CICS interface produces MVS SYSM dumps for some error conditions and MVS SDUMPs for other, more serious conditions. These dumps contain all the external CICS interface control blocks, as well as trace entries. You can use IPCS to format these dumps.

The MVS abends 0401 through 0415 can occur when you are running an external CICS interface job. For a description of these abends, see the *CICS/ESA Messages and Codes*.

A new user-replaceable program, DFHXCTRA, is available for use under the guidance of IBM service personnel. It is the equivalent of DFHTRAP used in CICS. It is invoked every time the external CICS interface writes a trace entry.

DFHXCTRA can perform one or all of the following actions:

1. Request the external CICS interface to write a trace entry on its behalf
2. Instruct the external CICS interface to take an SDUMP
3. Instruct the external CICS interface to skip writing the current trace entry to GTF
4. Instruct the external CICS interface to disable DFHXCTRA

The CICS-supplied sample version of DFHXCTRA performs all four of the above functions if it detects a trace entry that indicates that a FREEMAIN error occurred while trying to free an EXCI pipe control block.

The source for DFHXCTRA is supplied in CICS410.SDFHSRC. The parameter list passed to DFHXCTRA is defined in the copybook DFHXCTRD, which is supplied in CICS410.SDFHMAC.

Choosing between the EXEC CICS LINK or CALL interface

As illustrated in the various versions of the CICS-supplied sample client program, you can use both the CALL interface (all six commands) and the EXEC CICS LINK command in the same program, to perform separate requests. However, it is unlikely that you would want to do this in a production program.

Each form of the external CICS interface has its particular benefits.

- For low-frequency or single DPL requests you are recommended to use the EXEC CICS LINK command.

It is easier to code, and therefore less prone to programming errors.

However, each invocation of an EXEC CICS LINK command causes the external CICS interface to perform all the functions of the CALL interface, which results in unnecessary overhead.

- For multiple or frequent DPL requests from the same client program, you are recommended to use the EXCI CALL interface.

  This is more efficient, because you need perform the Initialize_User and Allocate_Pipe commands once only, at or near the beginning of your program, and the Deallocate_Pipe once on completion of all DPL activity. In between these functions, you can open and close the pipe as necessary, and while the pipe is opened, you can issue as many DPL calls as you want.

Compiling and link-editing external CICS interface client programs

This section discusses the following topics:

- The external CICS interface stub, DFHXCSTB
- The required linkage editor modes
- The CICS-supplied procedures for compiling and link-editing your client programs.

The external CICS interface stub, DFHXCSTB

All programs that use the external CICS interface to pass DPL requests to a CICS server region must include the CICS-supplied program stub, DFHXCSTB.

The stub intercepts all external CICS interface commands, whether they are EXCI CALL interface commands, or EXEC CICS LINK commands, and ensures they are passed to the appropriate external CICS interface routine for processing.

DFHXCSTB is a common stub, designed for inclusion in programs written in all the supported languages. It is supplied in the CICS410.SDFHEXCI library. This library also contains entries for DFHXCIE and DFHXCIS, which are aliases for DFHXCSTB.

To help you ensure that the stub is included, CICS provides a number of procedures, one for each language, which you can use for translating, compiling, and link-editing.

The required linkage editor modes

You must specify AMODE(31) for your EXCI client program.

The CICS-supplied procedures for compiling and link-editing client programs include the following parameters on the PARM statement of the linkage editor job step:

```
LNKPARM='AMODE(31),LIST,XREF'
```

The CICS-supplied procedures for the external CICS interface

CICS provides five procedures to enable you to translate, compile, and link-edit your client programs. These are:

DFHEXTAL The assembler procedure for assembler versions of client programs

DFHEXTDL The C procedure for C versions of client programs

DFHEXTPL The PL/I procedure for PL/I versions of client programs

DFHEXTVL The COBOL procedure for VS COBOL II versions of client programs.

To ensure that the EXCI stub is included with your client program, all these procedures include a step, COPYLINK, that unloads the stub into a temporary data set defined with a block length suitable for the linkage-editor. This temporary data set is then concatenated with the temporary data set containing your object program on the SYSLIN DD statement in the LKED step.

These procedures are supplied in the CICS410.SDFHPROC library. You are recommended to copy these to SYS1.PROCLIB or another suitable procedure library.

Job control language statements for running a client program

This section describes the JCL that you need to run an external CICS interface client program. The main points are as follows:

- You must include in the STEPLIB concatenation those libraries that contain the CICS-supplied external CICS interface modules needed by the job region and also the client program.

  The external CICS interface modules, including the sample client load module, are supplied in CICS410.SDFHEXCI.
- You are recommended to include a DD statement for SYSMDUMP. The external CICS interface uses SYSMDUMP for some error conditions.
- The REGION parameter must specify a large enough region size to allow for the size of the internal trace table specified by the TRACESZE parameter in the DFHXCOPT options table.
- Include a SYSPRINT or equivalent DD statement for any output from the client program.

Figure 14 on page 211 shows a sample job that you can use or modify to start a client program.

```
//EXCI    JOB (accounting_information),CLASS=A,TIME=1440,
//        USER=userid,PASSWORD=pswd,REGION=100M
//*-------------------------------------------------------------------*
//*    JCL to execute an external CICS interface client program   *
//*-------------------------------------------------------------------*
//          EXEC  PGM=pgmname
//STEPLIB   DD    DSN=CICS410.EXCI.LOADLIB,DISP=SHR
//          DD    DSN=CICS410.SDFHEXCI,DSIP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSMDUMP  DD    DSN=SYS1.SYSMDP00,VOL=SER=volid,SPACE=(CYL,(1,1)),
                  DISP=OLD,UNIT=3390
```

Figure 14. Sample job for starting a client program

Sample application programs

CICS provides a number of sample programs that are designed to help you in writing your own application programs. To help with writing programs that use the external CICS interface, CICS provides a sample MVS client program and a sample CICS server program.

The samples show you how to code client applications that use both the EXCI CALL interface and EXEC CICS LINK command.

Description of the sample applications

The sample external CICS interface programs are included on the CICS/ESA 4.1 distribution tape.

The sample MVS client program is provided in assembler language, VS COBOL II, C/370, and PL/I. The sample CICS server program is provided in assembler only. Assembler language programs are in source and executable form. COBOL, PL/I, and C/370 programs are provided in source form only. Each version of the client program has basically the same function, but programming methods vary somewhat according to the language used.

The sample programs, shown in Table 17 on page 212, are supplied in source form in CICS410.SDFHSAMP. The sample assembler server program is also supplied in executable form in CICS410.SDFHLOAD. The assembler client program is supplied in CICS410.SDFHEXCI.

Note: The assembler version of the client program uses BSAM, which requires the program to be link-edited in RMODE(24). The assembler source code includes the required RMODE(24) statement. Normally, EXCI client programs run AMODE(31),RMODE(ANY).

Table 17. The external CICS interface sample programs		
Language	**Name**	**Type of program**
Assembler	DFH$AXCC	Client program
Assembler	DFH$AXCS	Server program
COBOL	DFH0CXCC	Client program
PL/I	DFH$PXCC	Client program
C/370	DFH$DXCC	Client program

The sample client programs show you how to code a simple MVS client application using the EXCI CALL interface and the EXEC CICS LINK command.

Each version of the client is divided into three separate sections as follows:

1. The first section issues a single EXEC CICS LINK command to inquire on the state of the sample VSAM file, FILEA, in the target CICS system.

   If the file is in a suitable state, processing continues to sections two and three, which together provide complete examples of the use of the EXCI CALL interface.

2. The second section initiates a specific MRO connection to the target CICS system and, once the pipe is open, performs a series of calls that each retrieve a single sequential record from the sample VSAM file, until no more records are available.

3. The third section is a simple routine to close the target sample file once processing of the data is complete. It also terminates the MRO connection now that the link is no longer required.

Some of the parameters used on the EXCI CALL and EXEC CICS LINK commands in the client program need to be tailored for your own target CICS server region. Change these as required, then re-translate, compile (or assemble) and link-edit the program.

The variables and their values specified in the sample programs are given in Table 18.

Table 18. Parameters used in the sample client programs

Variable name in sample program	Value
TARGET_FILE	FILEA
TARGET_TRANSID	EXCI
TARGET_SYSTEM	DBDCCICS
TARGET_PROGRAM	DFH$AXCS
APPLICATION	BATCHCLI

The assembler version of the client program is supplied pregenerated in an executable form. All versions of the program accept a run-time parameter to specify the target server region APPLID. For the pregenerated assembler version this avoids you having to reassemble the program to specify the applid of your own CICS server region. You can also use the sample client program with different CICS regions without needing to modify the program each time.

Installing the EXCI sample definitions

Resource definitions that support the EXCI sample programs are included in the CICS system definition file (CSD) in groups DFH$EXCI and DFH$FILA.

Note that the sample definitions, while included in the CSD, are not included in the IBM-defined group list DFHLIST. Thus, if CICS is initialized with GRPLIST=DFHLIST, you must install the EXCI resource definition groups before using the samples. Alternatively, you can add the sample groups to your startup group list, so that they are installed automatically at system initialization.

The resource definition groups that must be installed are as follows:

DFH$EXCI This contains definitions for the sample server transaction, server program, EXCI connections and sessions.

Only one server program is included—in assembler language, called DFH$AXCS.

The sample application is designed to run the transaction EXCI, which is defined to invoke the DFHMIRS mirror program and references profile DFHCICSA. The required transaction definition for EXCI is included in the group.

Sample CONNECTION and SESSIONS definitions for specific and generic connections are included.

DFH$FILA This contains the definition for the supplied sample VSAM file, FILEA, which is referenced by the EXCI sample programs.

Once these are installed, you must ensure that interregion communication (IRC) is open. If IRC is not opened during CICS initialization, set it open using the CEMT SET IRC OPEN command.

Running the EXCI sample applications

If you want to use the COBOL, PL/I, or C/370 version of the EXCI client program, you must translate, compile, and link-edit the program into a suitable library.

You can use the sample JCL shown in Figure 14 on page 211 as a basis for creating your own batch job to run the client program.

If you use the pregenerated assembler version, you need to specify the APPLID of your target CICS server region as a parameter on the EXEC statement for the client program, as follows:

```
//*-------------------------------------------------------------*
//ASM       EXEC  PGM=DFH$AXCC,PARM='applid'
```

Results of running the EXCI sample applications

An example of the output produced by successful execution of the pregenerated assembler version of the client program, DFH$AXCC, is shown in Figure 15 on page 215.

If an error occurs while running the application, then, assuming the error is not severe, messages are written to the SYSPRINT output log displaying the reasons and/or return codes that cause processing to be aborted. Several examples of error-invoked output are shown in Figure 16, Figure 17, and Figure 18 on page 216.

```
*==================== EXCI Sample Client Program ============================*
*                                                                            *
*  EXEC Level Processor.                                                     *
*    Setting up the EXEC level call.                                         *
*    The Link Request has successfully completed.                            *
*    Server Response:                                                        *
*      The file is set to a browsable state.                                 *
*                                                                            *
*  CALL Level Processor.                                                     *
*    Initialize_User call complete.                                          *
*    Allocate_Pipe call complete.                                            *
*    Open_Pipe call complete.                                                *
*    The connection has been successful.                                     *
*      The target file follows:                                              *
*                                                                            *
*========================== Top of File =====================================*
 000102F. ALDSON           WARWICK, ENGLAND    9835618326 11 81$1111.11Y00007300
 000104S. BOWLER           LONDON,ENGLAND      1284629326 11 81$0999.99Y00007400
 000106B. ADAMS            CROYDON, ENGLAND    1948567326 11 81$0087.71Y00007500
 000111GENE BARLOWE        SARATOGA,CALIFORNIA 4612075301 02 74$0111.11Y00007600
 000762GEORGE BURROW       SAN JOSE,CALIFORNIA 2231212101 06 74$0000.00Y00007700
 000983H. L. L. CALL       WASHINGTON, DC      3451212021 04 75$9999.99Y00007800
 001222J.R.REYNOLDS        BOBLINGEN, GERMANY  7031555110 04 73$3349.99Y00007900
 001781HAROLD JAMES        SINDELFINGEN,GERMANY7031999021 06 77$0009.99Y00008000
 003210B.CREPIN            NICE, FRANCE        1234567026 11 81$3349.99Y00008100
 003214HUBERT C HERBERT    SUNNYVALE, CAL.     3411212000 06 73$0009.99N00008200
 003890PHILIPPE SMITH, JR  NICE, FRANCE        0000000028 05 74$0009.99N00008300
 004004STAN SMITH          DUBLIN, IRELAND     7111212102 11 73$1259.99N00008400
 004445S. GALSON           SOUTH BEND, S.DAK.  6121212026 11 81$0009.99N00008500
 004878D.C. CURRENT        SUNNYVALE, CALIF.   3221212010 06 73$5399.99N00008600
 005005J. S. LAVERENCE     SAN FRANCISCO, CA.  0000000101 08 73$0009.99N00008700
 005444JEAN LAWRENCE       SARATOGA, CALIF.    6771212020 10 74$0809.99N00008800
 005581JOHN ALDEN III      BOSTON, MASS.       4131212011 04 74$0259.99N00008900
 006016DR W. T. KAR        NEW DELHI, INDIA    7033121121 05 74$0009.88Y00009000
 006670WILLIAM KAPP        NEW YORK, N.Y.      2121212031 01 75$3509.88N00009100
 006968D. CONRAD           WARWICK, ENGLAND    5671382126 11 81$0009.88Y00009200
 007007BRIGITTE EICRN      STUTTGART, GERMANY  7031100010 10 75$5009.88N00009300
 007248B. C. WILLIAMSON    REDWOOD CITY, CALF. 3331212111 10 75$0009.88N00009400
 007779MRS. W. WELCH       SAN JOSE, CALIF.    4151212003 01 75$0009.88Y00009500
 100000G. NEADS            TORONTO, ONTARIO    0341512126 11 81$0010.00Y00009600
 111111C. MEARS            OTTAWA, ONTARIO     5121200326 11 81$0011.00Y00009700
 200000A. BONFIELD         GLASGOW,  SCOTLAND  6373829026 11 81$0020.00Y00009800
 222222J. WIEBERS          FRANKFURT, GERMANY  2003415126 11 81$0022.00Y00009900
 300000K. TRENCHARD        NEW YORK, U.S.      6473980126 11 81$0030.00Y00010000
 333333D. MYRING           CARDIFF, WALES      7849302026 11 81$0033.00Y00010100
 400000W. TANNER           MILAN, ITALY        2536373826 11 81$0040.00Y00010200
 444444A. FISHER           CALGARY, ALBERTA    7788982026 11 81$0044.00Y00010300
 500000J. DENFORD          MADRID, SPAIN       4445464026 11 81$0000.00Y00010400
 555555C. JARDINE          KINGSTON, N.Y.      3994442026 11 81$0005.00Y00010500
 600000F. HUGHES           DUBLIN, IRELAND     1239878026 11 81$0010.00Y00010600
 666666A. BROOKMAN         LA HULPE, BRUSSELS  4298384026 11 81$0016.00Y00010700
 700000A. MACALLA          DALLAS, TEXAS       5798432026 11 81$0002.00Y00010800
 777777D. PRYKE            WILLIAMSBURG, VIRG. 9187613126 11 81$0027.00Y00010900
 800000H. BRISTOW          WESTEND, LONDON     2423338926 11 81$0030.00Y00011000
 888888B. HOWARD           NORTHAMPTON, ENG.   2369163926 11 81$0038.00Y00011100
 900000D. WOODSON          TAMPA, FLA.         3566812026 11 81$0040.00Y00011200
 999999R. JACKSON          RALEIGH, N.Y.       8459163926 11 81$0049.00Y00011300
*========================== End of File =====================================*
*                                                                            *
*    Closing Dpl Request has been attempted.                                 *
*    Close_Pipe call complete.                                               *
*    Deallocate_Pipe call complete.                                          *
*                                                                            *
*================== End of EXCI Sample Client Program =======================*
```

Figure 15. Successful execution

```
*====================== EXCI Sample Client Program ============================*
*                                                                              *
*  EXEC Level Processor.                                                       *
*    Setting up the EXEC level call.                                           *
*    The Link Request has failed.  Return codes are;                           *
*        Resp = 00000088  Resp2 = 00000203  Abend Code:                        *
*    >>>> Aborting further processing <<<<                                     *
*                                                                              *
*=================== End of EXCI Sample Client Program ========================*
```

Figure 16. No CICS return code. The target CICS region specified by the client program is not found, or IRC was not opened.

```
*====================== EXCI Sample Client Program ============================*
*                                                                              *
*  EXEC Level Processor.                                                       *
*    Setting up the EXEC level call.                                           *
*    The Link Request has successfully completed.                              *
*    Server Response:                                                          *
*      The file could not be found.                                            *
*    >>>> Aborting further processing <<<<                                     *
*                                                                              *
*=================== End of EXCI Sample Client Program ========================*
```

Figure 17. No file found. The target file name to the server program was not found on the target CICS system.

```
*====================== EXCI Sample Client Program ============================*
*                                                                              *
*  EXEC Level Processor.                                                       *
*    Setting up the EXEC level call.                                           *
*    The Link Request has failed.  Return codes are;                           *
*        Resp = 00000088  Resp2 = 00000414  Abend Code:                        *
*    A message was received from the target CICS system:                       *
*                                                                              *
 DFHAC2001 04/29/93 16:43:03 IYAHZCAZ Transaction 'BAD_' is unrecognized.  Check
 that the transaction name is correct.
*                                                                              *
*    >>>> Aborting further processing <<<<                                     *
*                                                                              *
*=================== End of EXCI Sample Client Program ========================*
```

Figure 18. Incorrect transaction identifier. The target transid passed in the external CICS interface call is not defined on the target CICS system. Note the message received from the target CICS system.

Chapter 12. Access to CICS state data

This chapter describes the programming interface changes introduced in CICS/ESA 4.1 to allow access to CICS state data without the need to directly reference CICS control blocks. It covers the following topics:

- Overview
- Benefits of state data access
- Changes to CICS externals.

Overview

Access to CICS data through control blocks has been inhibited by the removal of macro support, and by removal of access to particular control blocks.

Access to CICS state data is now provided through enhancements to the system programming interface (SPI) and the exit programming interface (XPI), as follow:

- There are additional EXEC CICS INQUIRE commands:

  A browse function for REQID, giving you access to outstanding START, DELAY, and POST commands. (EXEC CICS INQUIRE REQID ...)

  A browse function for programs enabled at global and task-related user exits. (EXEC CICS INQUIRE EXITPROGRAM ...)
- Additional options on EXEC CICS INQUIRE and SET TERMINAL|NETNAME
- Additional keywords on EXEC CICS INQUIRE and SET SYSTEM
- A new XPI INQUIRE_SYSTEM function
- A new XPI SET_SYSTEM function
- A new XPI INQ_APPLICATION_DATA function.

Benefits of access to CICS state data

The improvements in access to CICS state data described in this chapter are designed to provide the following benefits:

- To enable CICS system programmers to obtain information about CICS through supported programming interfaces.
- To remove dependence on, and knowledge of, CICS control block structures.
- By means of the SPI enhancements, to provide a release-independent solution to enable CICS programs that access state data to work, unmodified, from release to release.

Changes to CICS externals

There are changes to some programming interfaces in CICS/ESA 4.1 to allow you to access CICS state data. These are:

- Changes to the system programming interface (SPI)
- Changes to the exit programming interface (XPI)

Changes to the system programming interface

There are some new options on the following INQUIRE and SET commands, and some new commands, as shown:

- INQUIRE CONNECTION (new options)
- INQUIRE EXITPROGRAM (new command)
- INQUIRE REQID (new command)
- INQUIRE SYSTEM (new options)
- SET SYSTEM (new options)
- INQUIRE TERMINAL (new options)
- SET TERMINAL (new options)

General-use programming interface

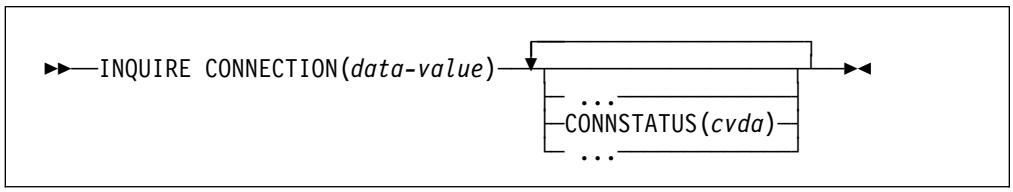
EXEC CICS INQUIRE CONNECTION command

Enhancements to this command enable you to obtain status information about MRO connections.

Function

Retrieves status information about a named connection to a remote system. The remote system can be another CICS region.

Syntax



INQUIRE CONNECTION options

CONNSTATUS(cvda)

returns a CVDA value identifying the status of the connection between CICS and a remote system. The remote system can be a logical unit (VTAM) or an MRO partner, identified by the name on the CONNECTION parameter.

The ACQUIRED and RELEASED values are common to both VTAM and MRO; the others are unique to APPC. CVDA values are:

ACQUIRED (VTAM and MRO)

The connection is acquired. The criteria for ACQUIRED for VTAM links are:

- The partner LU has been contacted.
- Initial CNOS exchange has been done.

The criteria for ACQUIRED for MRO links are:

- Both sides of the link are in service.
- Both sides of the link are successfully logged on to the CICS interregion communication program (DFHIRP).
- A connection request by each side has been successful for at least one session, and therefore each side can send and receive data.

AVAILABLE (APPC only)
The connection is acquired but there are currently no bound sessions because they were unbound for limited resource reasons.

FREEING (APPC only)
The connection is being released.

OBTAINING (APPC only)
The connection is being acquired. The connection remains in the OBTAINING state until all the criteria for ACQUIRED have been met.

RELEASED (VTAM and MRO)
The connection is released. Although the connection might be in service, it is not usable.

In the case of an MRO link, the released status can be caused by any one of a number of conditions. For example, it could be because the CICS region on the other side has not yet initialized, or not yet signed on to the CICS interregion communication program (DFHIRP); or it could be because CICS interregion communication has been closed on the other side, or the connection on the other side has been set out-of-service.

EXEC CICS INQUIRE EXITPROGRAM command

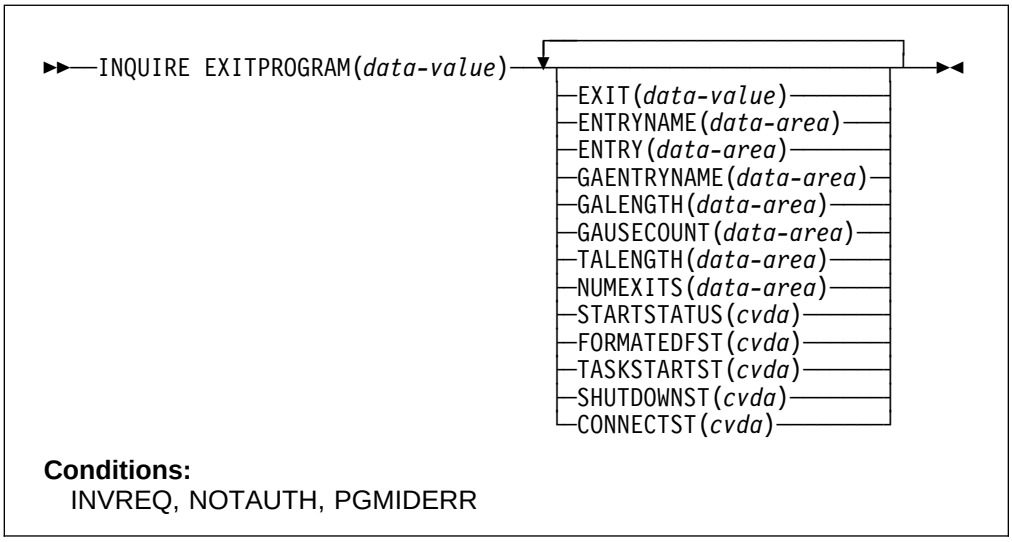
This is a new command in CICS/ESA 4.1 and is available in the following forms:

INQUIRE EXITPROGRAM(...)
INQUIRE EXITPROGRAM START
INQUIRE EXITPROGRAM(...) NEXT options...
INQUIRE EXITPROGRAM END

Function

Returns information about either global user exit programs or task-related user exit programs.

Syntax



Conditions:
INVREQ, NOTAUTH, PGMIDERR

The following commands allow you to browse all EXITPROGRAM definitions.

Browse EXITPROGRAM

INQUIRE EXITPROGRAM EXIT(*data-value*) START (global user exits)

INQUIRE EXITPROGRAM START (task-related user exits) (See note 2.)

INQUIRE EXITPROGRAM(data-area) NEXT

The options for the browse function are the same as for INQUIRE EXITPROGRAM.

INQUIRE EXITPROGRAM END

Notes:

1. The browse of user exit programs is done in the time sequence of the enable.
2. When you use the task-related user exit form of the browse commands, CICS returns information about both task-related and global user exit programs.
3. When you use the global user exit form of the browse commands, CICS returns information about global user exits only.

For information about browsing resources, see the *CICS/ESA System Programming Reference* manual.

INQUIRE EXITPROGRAM options

CONNECTST(cvda)

returns a CVDA value indicating the state of the connection between the exit and the external resource manager which it supports. This option applies only to the task-related user exits that connect CICS to DBCTL or to DB2. It enables you to determine whether the specified exit has connected to its resource manager, so that CICS tasks can safely issue API requests to the resource manager.

To inquire about the connection to DBCTL, use an EXITPROGRAM value of DFHDBAT and an ENTRYNAME value of DBCTL.

To inquire about the connection to DB2, use an EXITPROGRAM value of DSN2EXT1, with an ENTRYNAME of DSNCSQL, DSNCCMD, or DSNCIFC.

CVDA values are:

CONNECTED
: The task related user exit is connected to its external resource manager subsystem, and API requests can be issued.

NOTCONNECTED
: The task related user exit is not connected to its external resource manager subsystem, and therefore API requests cannot be issued.

NOTAPPLIC
: Not applicable. The user exit program is not a DBCTL or DB2 task-related user exit.

If the DBCTL or DB2 task-related user exits are not enabled, the INQUIRE command returns PGMIDERR. This also indicates that CICS is not connected to one of these resource manager subsystems.

Note: Use CONNECTST for the start status for DB2 and DBCTL task-related user exits, and not STARTSTATUS, which is for inquiring on other task-related exits, and global user user exits.

ENTRY(data-area)
returns a fullword binary field giving the entry address of the global or task-related user exit program identified by the ENTRYNAME and PROGRAM parameters.

ENTRYNAME(data-area)
returns the 8-character identifier of the global or task-related user exit program. The value can be the same as the name of the load module specified on the EXITPROGRAM parameter. A different value of ENTRYNAME is returned when the load module contains more than one exit program.

The value of ENTRYNAME is unique among the enabled ENTRYNAMEs at that exit. The combination of EXITPROGRAM and ENTRYNAME values uniquely identifies the user exit program.

EXIT(data-value)
specifies the 8-character identifier of the exit point upon which the inquiry is to be made. This option is provided for global user exit programs.

You must specify an EXIT value to inquire on global user exit programs.

Omit this option if the inquiry is for task-related user exits. The first task-related user exit returned by a browse is normally DFHEDP.

EXITPROGRAM(data-value)
specifies the 8-character name of the load module of the exit program.

Global user exits
On a global user exit browse operation, CICS returns the name of an exit program enabled at the specified exit point. For the first NEXT operation, CICS returns the name of the first to be enabled at that point. On each subsequent browse, CICS returns the next exit program in time sequence.

Task-related user exits
On a task-related browse operation, CICS returns information about ***all*** exit programs in the system. You can determine which are task-related by testing the value returned on the NUMEXITS parameter. If the value is zero, the program is a task-related user exit program.

FORMATEDFST(cvda)
returns a CVDA value identifying whether FORMATEDF was specified on the ENABLE command. CVDA values are:

FORMATEDF
The screen is modified when EDF is on

NOFORMATEDF
The screen is not modified when EDF is on

NOTAPPLIC
Not applicable

This option is ignored for global user exits.

GAENTRYNAME(data-area)
returns an 8-character string giving the name of the currently enabled global, or task-related, user exit program that owns the global work area being used by the exit program named on the ENTRYNAME or EXITPROGRAM parameter.

This value is returned only when the exit program being inquired upon is using a global work area owned by another exit program.

The value returned is either the load module name of the program (the EXITPROGRAM value), or, if it has an ENTRYNAME, the ENTRYNAME value.

GALENGTH(data-area)
returns a halfword binary field giving the length of the global work area for this exit program.

GAUSECOUNT(data-area)
returns a halfword binary field giving the total number of global or task related user exit programs that are using the global work area owned by this exit program. This count includes the owning exit program.

NUMEXITS(data-area)
returns a halfword binary field giving the number of global user exit points at which the exit program is enabled.

SHUTDOWNST(cvda)
returns a CVDA value giving whether the task-related user exit is invoked when a CICS shutdown occurs. CVDA values are:

NOSHUTDOWN
The task-related user exit is not invoked when a CICS shutdown occurs

NOTAPPLIC
Not applicable

SHUTDOWN
The task-related user exit is invoked when a CICS shutdown occurs

This option is ignored for global user exits.

STARTSTATUS(cvda)
returns a CVDA value identifying whether the exit program is available for execution. CVDA values are:

STARTED
The exit program is available for execution; that is, the START option on an EXEC CICS ENABLE command is still valid.

STOPPED
The exit program is not available for execution; that is, the START option has not been issued, or has been revoked by the STOP option on an EXEC CICS DISABLE command.

TALENGTH(data-area)
returns a halfword binary field giving the length of a local (task-related) work area. The work area is released at the end of the task for which it was created.

This option is ignored for global user exits.

TASKSTARTST(cvda)
returns a CVDA value identifying whether the task-related user exit program is set to be invoked automatically at the start and end of every task. CVDA values are:

NOTAPPLIC
Not applicable.

NOTASKSTART
The exit program is not set for invocation at the start and end of every task.

TASKSTART
The exit program is set for invocation at the start and end of every task.

This option is ignored for global user exits.

INQUIRE EXITPROGRAM conditions

INVREQ
The exit point specified does not exist (RESP2=3).

NOTAUTH
The use of this command is not authorized (RESP2=100).

NOTAUTH
The access you have requested to this user exit program is not authorized (RESP2=101).

PGMIDERR
is returned, with RESP2=1, if:

- The load module named on the EXITPROGRAM parameter is not defined to CICS, or
- The load module is not in the load library, or
- The load module has been disabled, or
- The EXITPROGRAM is not enabled, or
- The EXIT parameter is missing for an inquire on a global user exit program.

EXEC CICS INQUIRE REQID command

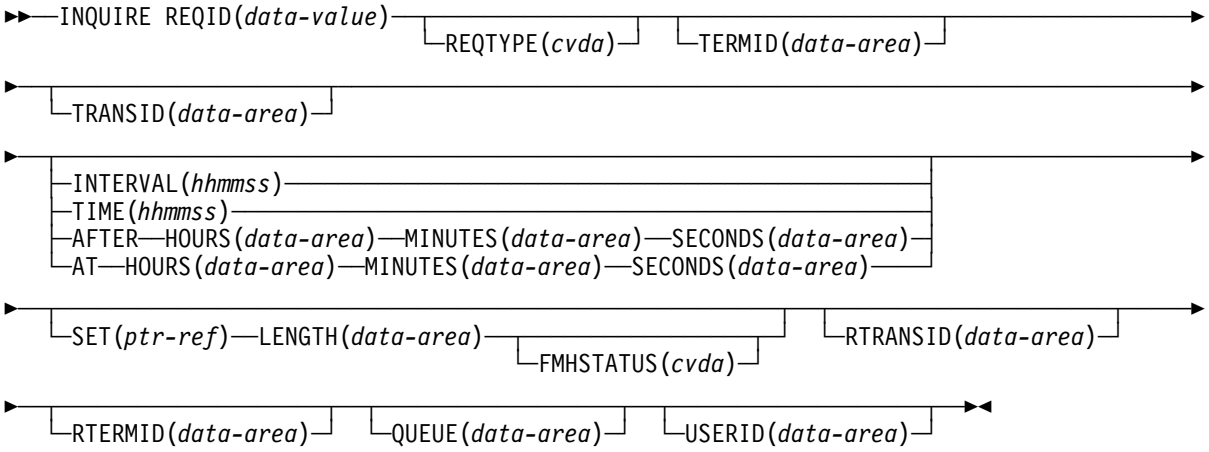
This is a new command, available in the following forms:

- INQUIRE REQID()
- INQUIRE REQID() START
- INQUIRE REQID() NEXT options...
- INQUIRE REQID() END

Function

To retrieve information about a queued request (whose REQID is known).

Syntax



Conditions:
INVREQ, NOTAUTH, NOTFND

This command allows you to recover the data associated with a specific queued request (whose REQID is known). You can then delete (CANCEL) the original request and optionally add a new request with modified data (for example, using EXEC CICS START).

This command takes a REQID value and finds the first matching request in the queue. You specify the information to return about this request by including the relevant options.

You can use the following browse commands to obtain access to outstanding EXEC CICS START, DELAY, and POST commands.

Browse REQID

INQUIRE REQID START

INQUIRE REQID(data-area) NEXT

The options for the browse function are the same as for INQUIRE REQID.

INQUIRE REQID END

For information about browsing resources, see the *CICS/ESA System Programming Reference* manual.

INQUIRE REQID options

AFTER
specifies that the values of HOURS, MINUTES and SECONDS refer to an **interval** of time that is to elapse before this queued request expires.

AT
specifies that the values of HOURS, MINUTES and SECONDS refer to the actual **time** at which this queued request expires.

FMHSTATUS(cvda)
returns a CVDA value identifying whether the data associated with the queued request contains function management headers. CVDA values are:

FMH
The data associated with the queued request contains a function management header.

NOFMH
The data associated with the queued request does not contain a function management header.

NOTAPPLIC
Not applicable, because there is no data associated with the request, or the REQTYPE CVDA value is not START.

HOURS(data-area)
returns a fullword binary field giving the hours portion of the expiry time specified with the AT or the AFTER option:

- If you specify AT, the hours are the number of hours from the previous midnight.
- If you specify AFTER, the hours are the number of hours from the time your INQUIRE command is processed.

INTERVAL(data-area)
returns an 8-digit packed decimal field (4 bytes) giving the interval until expiry of the queued request. This time is in the format **0hhmmss+**.

LENGTH(data-area)
returns a halfword binary field giving the length of the data associated with the queued request. If no data is supplied, or if the REQTYPE is not START, the value is zero.

MINUTES(data-area)
returns a fullword binary field giving the minutes portion of the expiry time specified with either the AT or AFTER options.

QUEUE(data-area)
returns an 8-byte string giving the queue name associated with the queued request. This corresponds to the value on the QUEUE parameter of the START command that created the queued request. If there is no queue associated with the START, or if the REQTYPE is not START, CICS returns blanks.

The data is obtained from temporary storage (TS) and the read can fail either because of an I/O error or because the TS queue has been deleted. In either case, CICS returns the INVREQ condition.

REQID(data-value)
specifies the 8-byte identifier of a request queued in the system. If the value is not associated with any request, NOTFND is returned.

REQTYPE(cvda)
returns a CVDA value identifying the type of queued request that the inquiry command relates to. CVDA values are:

DELAY
The queued request was issued by a DELAY command.

POST
The queued request was issued by a POST command.

START
The queued request was issued by a START command.

RTERMID(data-area)
returns a 4-byte string giving the TERMID associated with the queued request. This corresponds to the value in the RTERMID parameter of the START command that created the queued request. If no RTERMID is associated with the request, or if the REQTYPE is not START, CICS returns blanks.

The data is obtained from temporary storage (TS) and the read can fail, either because of an I/O error or because the temporary storage queue has been deleted. In either case, the INVREQ condition is returned.

RTRANSID(data-area)
returns a 4-byte string giving the transaction identifier associated with the queued request. This corresponds to the value in the RTRANSID parameter of the START command that created the queued request. If no RTRANSID is associated with the request, or if the REQTYPE is not START, the field is set to blanks.

The data is obtained from temporary storage (TS) and the read can fail, either because of an I/O error or because the TS queue has been deleted. In either case, the INVREQ condition is returned.

SECONDS(data-area)
returns a fullword binary field giving the seconds portion (0-59) of the expiry time specified with the AT or the AFTER option.

SET(ptr-ref)
returns a pointer reference set to the address of an area of storage containing data associated with the queued request. This is the data supplied to the original START transaction using the FROM parameter. If no data is supplied, or if the REQTYPE is not START, the pointer is set to NULL (the field is set to blanks).

The data is obtained from temporary storage (TS) and the read can fail, either because of an I/O error or because the temporary storage queue has been deleted. In either case, the INVREQ condition is returned.

TERMID(data-area)
returns a 4-byte string giving the TERMID associated with the queued request. This corresponds to the value on the TERMID parameter of the START command that created the queued request. If no terminal is associated with this request, or if the REQTYPE is not START, the field is set to blanks.

TIME(data-area)
returns an 8-digit packed decimal string (4 bytes) giving the expiry time associated with the queued request. This is the actual time of day, in the format **0hhmmss+**.

TRANSID(data-area)
returns a 4-character string giving the transaction identifier associated with the queued request. This corresponds to the value on the TRANSID parameter of the START command that created the queued request. If the REQTYPE is not START, the field is set to blanks.

USERID(data-area)
returns an 8-byte string giving the userid of the user associated with the task that created this queued request. If the REQTYPE is not START, the field is set to blanks.

INQUIRE REQID conditions

INVREQ
Occurs in any of the following situations:

- TS GET has failed.
- An I/O error occurred when reading the temporary storage queue (RESP2=3), or the temporary storage no longer exists (RESP2=4).

NOTAUTH
The use of this command is not authorized (RESP2=100).

NOTFND
An I/O error occurred when reading the temporary storage queue (RESP2=3).

NOTFND
The temporary storage queue no longer exists (RESP2=4).

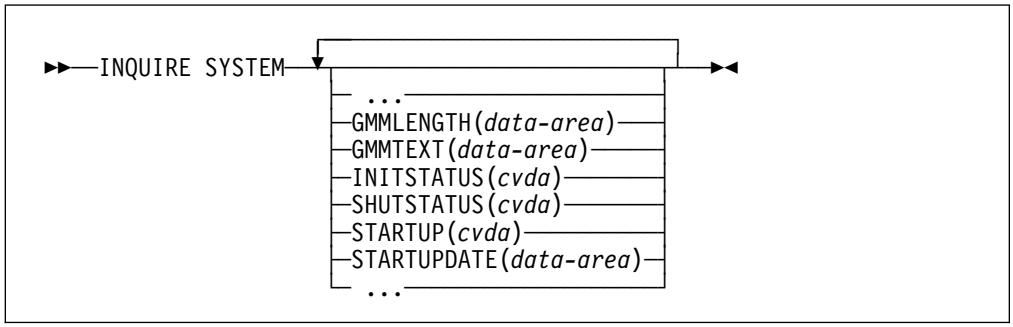
EXEC CICS INQUIRE SYSTEM command

There are extensions to the INQUIRE SYSTEM command.

Function

Retrieves information about the local CICS system.

Syntax



INQUIRE SYSTEM options

GMMLENGTH(data-area)
returns a halfword binary field giving the length of the CICS “good morning” message text, which can be up to a maximum of 246 bytes.

GMMTEXT(data-area)
returns a character string, of up to 246 bytes, of the text of the CICS “good morning” message.

INITSTATUS(cvda)
returns a CVDA value identifying the initialization status of the local CICS system. CVDA values are:

INITCOMPLETE
CICS initialization is complete.

SECONDINIT
Second stage of CICS initialization. This initialization status is detectable only from a first-stage PLTPI program or a global user exit program.

THIRDINIT
Third stage of CICS initialization. This initialization status is detectable only from a PLTPI program or a global user exit program.

SHUTSTATUS(cvda)
returns a CVDA value identifying the shutdown status of the local CICS system. CVDA values are:

CONTROLSHUT
CICS is performing a controlled shutdown; that is, a normal shutdown with a warm keypoint.

NOTAPPLIC
SHUTSTATUS is not applicable because CICS is not in shutdown mode.

SHUTDOWN
CICS is performing an immediate shutdown.

STARTUP(cvda)
returns a CVDA value identifying the type of startup performed by CICS for this run. CVDA values are:

COLDSTART
CICS performed a cold start, either because it was explicitly specified on the system initialization parameter, or CICS forced a cold start because of the state of the CICS global catalog.

EMERGENCY
CICS performed an emergency restart because the previous run of did not shutdown normally with a warm keypoint.

WARMSTART
CICS performed a warm restart following the normal shutdown of the previous run.

STARTUPDATE(data-area)
returns an 8-character decimal value giving the date the CICS system started in the packed decimal form **00yyddd+** where **yy** is the year, **ddd** is the days and **+** is the sign character.

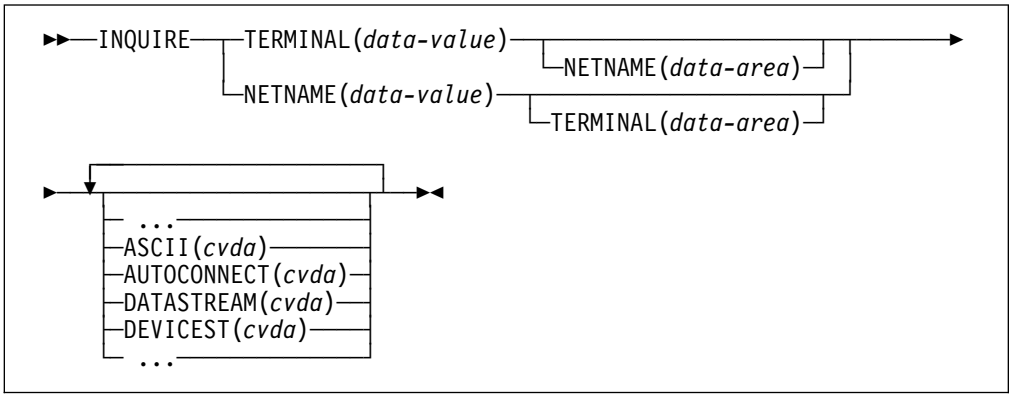
EXEC CICS INQUIRE TERMINAL | NETNAME command

There are new parameters to the SET TERMINAL (or SET NETNAME) command.

Function

Returns information about a named terminal.

Syntax



INQUIRE TERMINAL options

ASCII(cvda)
returns a CVDA value identifying the type of ASCII data stream being used. CVDA values are:

ASCII7 A 7-bit ASCII data stream.

ASCII8 An 8-bit ASCII data stream.

NOTAPPLIC Not applicable.

AUTOCONNECT(cvda)
returns a CVDA value identifying whether sessions with this terminal are to be established (bound) when CICS is initialized or whenever communication with VTAM is started.

CVDA values are:

ALLCONN The same as AUTOCONN. This value can be used to indicate that the associated MODENAME is specified as ALLCONN.

AUTOCONN
CICS binds associated sessions.

NONAUTOCONN
CICS does not bind associated sessions.

DATASTREAM(cvda)
returns a CVDA value identifying the device data stream type. CVDA values are:

DS3270 3270 data stream.

NOTAPPLIC
Not applicable.

SCS Standard character string.

DEVICEST(cvda)
returns a CVDA value identifying whether the device is busy. CVDA values are:

BUSY The device is busy.

NOTBUSY
The device is not busy.

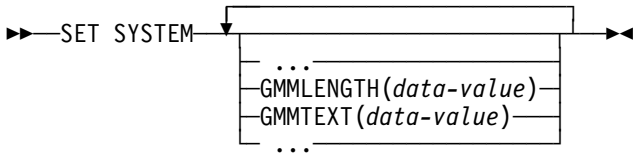
EXEC CICS SET SYSTEM command

There are additions to the SET SYSTEM command.

Function

Change the value of some of the system attributes.

Syntax



Conditions:
LENGERR

SET SYSTEM options

GMMLENGTH(data-value)
specifies, as a halfword binary variable, the length of the CICS “good morning” message text, up to a maximum value of 246 bytes.

GMMTEXT(data-area)
specifies the text of the CICS “good morning” message. The text can be up to 246 characters in length.

SET SYSTEM conditions

LENGERR

The length of GMMTEXT is greater than 246 or less than zero (RESP2=20).

End of General-use programming interface

Changes to the exit programming interface (XPI)

Product-sensitive programming interface

There are three new macro function calls added to the exit programming interface (XPI) to provide global user exit programs with access to some CICS state data. These are:

- INQUIRE_SYSTEM
- SET_SYSTEM
- INQ_APPLICATION_DATA

The INQUIRE_SYSTEM call

The INQUIRE_SYSTEM call gives you access to CICS system data in the AP Domain.

INQUIRE_SYSTEM

```
DFHSAIQX [CALL,]
   [CLEAR,]
   [IN,
   FUNCTION(INQUIRE_SYSTEM),]
   [GMMTEXT(name4),]
   [OUT,
   [AKP(name4 | *),]
   [CICSREL(name4 | *),]
   [CICSSTATUS(name1 | *),]
   [CICSSYS(name1 | *),]
   [CWA(name4 | (Rn) | *),]
   [CWALENGTH(name2 | *),]
   [DATE(name4|*),]
   [DTRPRGRM(name8 | *),]
   [GMMLENGTH(name2 | *),]
   [GMMTRANID(name4 | *),]
   [INITSTATUS(name1 | *),]
   [JOBNAME(name8 | *),]
   [OPREL(name4 | *),]
   [OPSYS(name1 | *),]
   [SECURITYMGR(name1 | *),]
   [SHUTSTATUS(name1 | *),]
   [STARTUP(name1 | *),]
   [XRFSTATUS(name1 | *),]
   [STARTUPDATE(name4| *),]
   [TIMEOFDAY(name4| *),]
   RESPONSE (name1 | * ),
   REASON (name1 | * )]
```

FUNCTION(INQUIRE_SYSTEM): The INQUIRE_SYSTEM function is provided on the DFHSAIQX macro call. Its purpose is to enable you to inquire on specified system attributes. For information about the XPI argument types, see the *CICS/ESA Customization Guide*.

AKP(name4 | *)

The activity keypointing frequency of the CICS region.

name4 The name of a 4-byte location that is to receive the frequency value

CICSREL(name4 | *)

The release under which the CICS region is running.

name4 The name of a 4-byte location that is to receive the release characters as hexadecimal values.

CICSSTATUS(name1 | *)

indicates, in a 1-byte location (*name*), the status of the CICS region. The equated values for the CICS status are as follows:

INITIALIZING The CICS region is initializing

ACTIVE The CICS region is active and ready to receive work.

FIRSTQUIESCE The CICS region is shutting down, and is in the first stage of quiescing.

FINALQUIESCE The CICS region is shutting down, and is in the final stage of quiescing.

CICSSYS(name1 | *)
The operating system for which the running CICS has been built.

name1 The name of a 1-byte area that is to receive the hexadecimal character of the operating system. A value of “X” represents MVS/ESA.

CWA(name4 | (Rn) | *)
The address of the common work area.

name4 The name of a 4-byte field that is to receive the address of the CWA.

(Rn) A register to receive the address of the CWA.

CWALENGTH(name2 | *)
The length in bytes of the CWA.

name2 The name of a 2-byte field that is to receive the length of the CWA.

DATE(name4 | *)
Today’s date in packed-decimal form (4-bytes **00yyddd**c where **yy**=years, **ddd**=days, **c** is the sign).

name4 The name of a 4-byte location that is to receive the date.

DTRPRGRM(name8 | *)
The name of the dynamic transaction routing program.

name8 The name of an 8-byte area that is to receive the name of the dynamic transaction routing program.

GMMLENGTH(name2 | *)
The length in bytes of the “good morning” message.

name2 The name of a 2-byte area that is to receive the length of the good morning message.

GMMTEXT(name4)
The address of an area of storage, at least 246 bytes in length and owned by the caller, into which CICS is to return the good morning message.

name4 The address of an area of storage that is to receive the good morning message.

Note: The GMMTEXT parameter must follow the IN statement as an input parameter.

GMMTRANID(name4 | *)
The transaction identifier of the CICS good morning transaction.

name4 The name of a 4-byte area that is to receive the CICS good morning transaction id.

INITSTATUS(name1 | *)
indicates, in a 1-byte location (*name1*), the stage reached during CICS initialization.

The equated values are:

FIRSTINIT The first stage of CICS initialization.

SECONDINIT The second stage of CICS initialization. This stage corresponds to the period when first phase PLTPI programs are run; that is those programs in a PLT that are defined **before** the DFHDELIM statement.

THIRDINIT The third stage of CICS initialization. This stage corresponds to the period when second phase PLTPI programs are run; that is those programs in a PLT that are defined **after** the DFHDELIM statement.

INITCOMPLETE
CICS initialization is complete.

JOBNAME(name8 | *)
The 8-character MVS job name under which the CICS region is running.

name8 The name of a 8-byte area that is to receive the MVS job name.

OPREL(name2 | *)
The release number of the currently running operating system

name2 The name of a 2-byte area that is to receive, as a half-word binary value, the release number of the operating system.

OPSYS(name1 | *)
The type of operating system on which the CICS regions is running.

name1 The name of a 1-byte area that is to receive the hexadecimal character of the operating system on which CICS is running. A value of “X” represents MVS/ESA.

SECURITYMGR(name1 | *)
indicates, in a 1-byte location (*name*), whether an external security manager is active in the CICS region, or whether security is not being used.

The equated values are:

EXTSECURITY CICS is using an external security manager (for example, RACF).

NOSECURITY. Security is not in use in the CICS region—SEC=NO is specified as a system initialization parameter.

SHUTSTATUS(name1 | *)
indicates, in a 1-byte location (**name1**), the shutdown status of the CICS region.

The equated values are:

SHUTDOWN CICS is performing an immediate shutdown.

CONTROLSHUT CICS is performing a controlled shutdown; that is, a normal shutdown with a warm keypoint.

NOTSHUTDOWN CICS is not in shutdown mode.

STARTUP(name1 | *)
indicates, in a 1-byte location (*name1*), the type of start up the CICS region performed.

The equated values are:

COLDSTART CICS performed a cold start, either because it was explicitly specified on the system initialization parameter, or CICS forced a cold start because of the state of the CICS global catalog.

EMERGENCY CICS performed an emergency restart because the previous run of did not shutdown normally with a warm keypoint.

WARMSTART CICS performed a warm restart following the normal shutdown of the previous run.

STARTUPDATE(name4 | *)
The start-up-date of this CICS region, in packed decimal form (4-bytes **00yydddc** where **yy**=years, **ddd**=days, **c** is the sign).

name4 The name of a 4-byte location that is to receive the startup date of this CICS system.

TIMEOFDAY(name4 | *)
The current time-of-day in packed decimal form (4-bytes **hhmmsstc** where **hh**=hours, **mm**=minutes, **ss**=seconds, **t**=tenths of a second, and **c** is the sign).

name4 The name of a 4-byte location that is to receive the time.

XRFSTATUS(name1 | *)
indicates, in a 1-byte location (*name1*), whether the CICS region started as an active or as an alternate CICS.

The equated values are:

PRIMARY The CICS region started as an active CICS.

TAKEOVER The CICS region was the alternate CICS, started with the START=STANDBY system initialization parameter.

These values are valid only if CICS started with the system initialization parameter XRF=YES specified.

RESPONSE and REASON values for INQUIRE_SYSTEM

RESPONSE	*REASON*
OK	None
INVALID	INVALID_FUNCTION
EXCEPTION	UNKNOWN_DATA
	LENGTH_ERROR
DISASTER	INQ_FAILED
PURGED	None

SET_SYSTEM

The SET_SYSTEM call allows you to set CICS system data values in the AP Domain.

SET_SYSTEM

```
DFHSAIQX [CALL,]
   [CLEAR,]
   [IN,
   FUNCTION(SET_SYSTEM),
   [AKP(name4 | (Rn) ),]
   [DTRPRGRM(name8 | string | 'string'),]
   [GMMLENGTH(name2 | (Rn) | expression),]
   [GMMTEXT(name8 | (Rn)),]
   [OUT,
   RESPONSE (name1 | * ),
   REASON (name1 | * )]
```

FUNCTION(SET_SYSTEM): The SET_SYSTEM function is provided on the DFHSAIQX macro call. Its purpose is to enable you to set specified system attributes. For information about the XPI argument types, see the *CICS/ESA Customization Guide*.

AKP(name4 | (Rn))
The activity keypointing frequency of the CICS region.

name4 The name of a 4-byte location that contains the new frequency value.

(Rn) A register that contains the new frequency value.

DTRPRGRM(name8 | string | 'string')
The name of the dynamic transaction routing program

name8 The name of an 8-byte area that contains the name of the dynamic transaction routing program.

string A string of character, without intervening blanks, that defines the name of the dynamic transaction routing program being set.

'string' A string of character without intervening blanks. If you want to document a name (label) in your program, use this form.

GMMLENGTH(name2 | (Rn))
The length of the new "good morning" message text that is in the area of storage referenced by the GMMTEXT parameter.

name2 The name of a 2-byte area that contains, as a half-word binary value, the length of the new good morning message.

(Rn) A register that contains the length of the new good morning message.

GMMTEXT(name4 | (Rn))
specifies a location, or register, that contains the address of an area of storage (up to a maximum of 246 bytes) that contains the new good morning message.

name4 The name of a 4-byte location that contains the address of the storage area that contains the good morning message.

(Rn) A register that contains the address of the storage area that contains the good morning message.

RESPONSE and REASON values for SET_SYSTEM:

RESPONSE	*REASON*
OK	None
INVALID	INVALID_FUNCTION
EXCEPTION	AKP_SIZE_ERROR
	NO_KEYPOINT
DISASTER	SET_FAILED
PURGED	None

The INQ_APPLICATION_DATA call

The INQ_APPLICATION_DATA call enables you to inquire on application system data in the AP Domain.

INQ_APPLICATION_DATA

```
DFHAPIQX [CALL,]
   [CLEAR,]
   [IN,
   FUNCTION(INQ_APPLICATION_DATA),]
   [OUT,
   [DSA(name4 | (Rn) | * ),]
   [EIB(name4 | (Rn) | * ),]
   [RSA(name4 | (Rn) | * ),]
   [SYSEIB(name4 | (Rn) | * ),]
   [TCTUA(name4 | (Rn) | * ),]
   [TCTUASIZE(name4 | * ),]
   [TWA(name4 | (Rn) | * ),]
   [TWASIZE(name4 | (Rn) | * ),]
   RESPONSE (name1 | * ),
   REASON (name1 | * )]
```

FUNCTION(INQ_APPLICATION_DATA): The INQ_APPLICATION_DATA function is provided on the DFHAPIQX macro call. Its purpose is to enable you to inquire on task-related control blocks, such as the transaction work area (TWA) and EXEC interface block (EIB).

Zero values for some addresses

When you use this XPI call from AP domain global user exits that support EXEC CICS commands (such as XFCREQ, XTSEREQ, and so on) CICS returns zero values for the following options:

EIB
SYSEIB
DSA
RSA

DFHAPIQX calls cannot be used in any exit program invoked from any global user exit points in the following domains or control program:

- Statistics domain
- Monitor domain
- Dump domain
- Dispatcher domain
- Journal control program

For information about the XPI argument types, see the *CICS/ESA Customization Guide*.

DSA(name4 | (Rn | *)
The head of the chain of dynamic storage used by application programs to make them reentrant (for example, for assembler programs, the DFHEISTG storage).

name4 The name of a 4-byte area that is to receive the address of the head of the dynamic storage chain.

(Rn) A register that is to receive the DSA address.

***** The parameter list itself, in name APIQ_DSA, is used to hold the address.

EIB(name4 | (Rn) | *)
The address of the EXEC interface block (EIB) for the current task.

name4 The name of a fullword area that is to receive the address of the EIB.

(Rn) A register that is to receive the address of the EIB.

***** The parameter list itself, in name APIQ_EIB, is used to hold the address.

RSA(name4 | (Rn | *)
The address of the register save area for the current task.

name4 The name of a fullword area that is to receive the address of the register save area.

(Rn) A register that is to receive the address of the register save area.

***** The parameter list itself, name APIQ_RSA, is used to hold the address.

SYSEIB(name4 | (Rn) | *)
The address of the system EXEC interface block of the current task.

name4 The name of a fullword area that is to receive the address of the system EXEC interface block.

(Rn) A register that is to receive the address of the system EXEC interface block.

***** The parameter list itself, name APIQ_SYSEIB, is used to hold the address.

TCTUA(name4 | (Rn) | *)
The address of the terminal control table user area (TCTUA) for the current task.

name4 The name of a fullword area that is to receive the address of the TCTUA.

(Rn) A register that is to receive the address of the TCTUA.

***** The parameter list itself, name APIQ_TCTUA, is used to hold the address.

TCTUASIZE(name4 | (Rn) | *)
The length in bytes of the TCTUA for the current task.

name4 The name of a 4-byte area that is to receive the length in bytes of the TCTUA.

(Rn) A register that is to receive the length of the TCTUA.

***** The parameter list itself, name APIQ_TCTUASIZE, is used to hold the length of the TCTUA.

TWA(name4 | (Rn) | *)
The address of the transaction work area.

name4 The name of a fullword area that is to receive the address of the TWA.

(Rn) A register that is to receive the address of the TWA.

***** The parameter list itself, name APIQ_TWA, is used to hold the address of the TWA.

TWASIZE(name4 | (Rn) | *)
The length, in bytes, of the transaction work area (TWA).

name4 The name of a 4-byte area that is to receive the length, in bytes, of the TWA.

(Rn) A register that is to receive the length of the TWA.

***** The parameter list itself, name APIQ_TWASIZE, is used to hold the length of the TWA.

RESPONSE and REASON values for INQ_APPLICATION_DATA:

RESPONSE	*REASON*
OK	None
EXCEPTION	DPL_PROGRAM
	NO_TRANSACTION_ENVIRONMENT
	TRANSACTION_DOMAIN_ERROR
DISASTER	ABEND
	LOOP
	INQ_FAILED
INVALID	INVALID_FUNCTION
KERNERROR	None
PURGED	None

End of Product-sensitive programming interface

Chapter 13. Cancel start requests

This chapter describes the improved handling of the automatic initiate descriptors (AIDs) by providing the ability to cancel AIDs for terminals and connections. It covers the following topics:

- Overview
- Benefits of canceling start requests
- Changes to CICS externals
- Problem determination.

Overview

This new function allows you to cancel AIDs on the AID chain. These AIDS could, for example, represent:

- Scheduled requests to start new transactions, and
- Allocated requests from existing transactions waiting for a named terminal facility.

In this context a “named terminal facility” means:

- A named terminal, or
- A named LU6.2, MRO, or LU6.1 connection.

You can now cancel AIDs using one of the following commands:

- CEMT SET TERMINAL or EXEC CICS SET TERMINAL commands, or
- CEMT SET CONNECTION or EXEC CICS SET CONNECTION commands.

Note: The support provided does NOT allow you to selectively cancel AIDs. You can cancel all eligible AIDs for a particular terminal or connection at the time of issuing the command, but any subsequent AIDs are not canceled.

Perhaps the most frequently-used type of schedule request occurs as the result of an EXEC CICS START TERMID command issued by a user program. When the time interval (if any) expires, CICS creates a scheduled request for the specified terminal.

The EXEC CICS START TERMID command can additionally specify that data is to be associated with the request, for later retrieval by the started task (the data being stored in the interim on a temporary storage queue). When such a request is canceled, the associated temporary storage queue is deleted.

A new global user exit is provided. This is invoked when an AID, resulting from an EXEC CICS START TERMID FROM(data) command executed in a local system, is canceled. This includes starts with data from other commands, for example, FEPI START.

Benefits of canceling start requests

You can now cancel scheduled requests using a documented programming interface, without the need to access CICS control blocks.

The general-use programming interface solution provided for this purpose ensures upward compatibility of any user application programs written to perform the cancel task.

Removing any dependency on CICS control blocks improves product reliability.

Changes to CICS externals

There are changes to the following interfaces in CICS/ESA 4.1 to support the cancelation of AIDs:

- Changes to the system programming interface
- Changes to CICS-supplied transactions
- Changes to global user exits

Changes to the system programming interface

A new CANCEL option is provided on both the EXEC CICS SET TERMINAL and EXEC CICS SET CONNECTION commands.

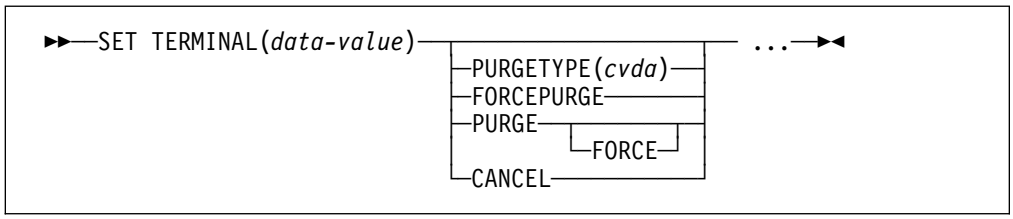
EXEC CICS SET TERMINAL command

General-use programming interface

Function

To change some terminal attributes, and cancel outstanding AIDs.

Syntax



SET TERMINAL options

PURGETYPE(cvda)

specifies whether transactions or queued automatic initiate descriptors (AIDs) associated with the named terminal can be purged. The CVDA values are:

CANCEL

Automatic initiate descriptors (AIDs) queuing for the specified terminal are to be canceled.

AIDs representing scheduled and allocated requests waiting in the local CICS system for the specified terminal are canceled. However, transient

data AIDs with an associated triggered task already started are not canceled. These AIDs can be removed by purging the associated task.

When a canceled scheduled request is found to have a precursor in a remote CICS system—that is, the AID was originally scheduled in a remote system—then that remote AID is also canceled, asynchronously.

Note that CICS returns a NORMAL response even if the terminal name is not defined to CICS. In this event, message DFHTF0100 indicates that no AIDs have been canceled.

SET TERMINAL conditions

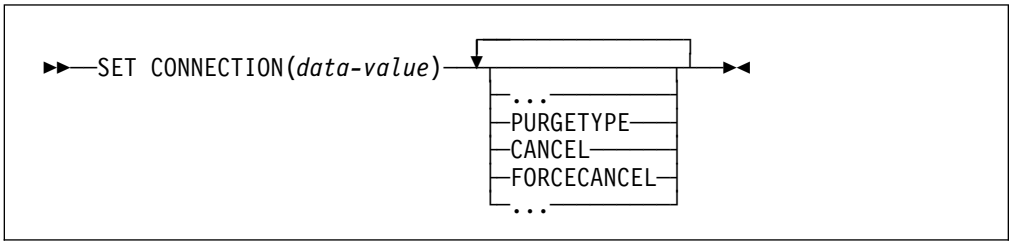
Condition	*RESP2*	*Meaning*
INVREQ	57	Other SET parameters were included with the CANCEL option.
NORMAL	58	AIDs are successfully canceled.

EXEC CICS SET CONNECTION command

Function

To change some connection attributes, and cancel outstanding AIDs.

Syntax



SET CONNECTION options

CONNECTION(data-value)

specifies, as a 4-character field, the LU6.2, MRO, or LU6.1 connection name.

PURGETYPE(cvda)

specifies how transactions on this connection are to be purged. CVDA values are:

CANCEL

specifies that queued automatic initiate descriptors (AIDs) are to be canceled for the specified connection.

AIDs representing scheduled and allocated requests waiting in the local CICS system for the specified connection are canceled. However, transient data AIDs with an associated triggered task already started are not canceled.

In addition, the following CICS system AIDs are not purged unless FORCECANCEL is specified.

Table 19. System AIDs requiring FORCECANCEL for removal	
Remote delete AIDs	
Remote scheduler AIDs	(CRSR)
LU6.2 service manager 1 AIDs	(CLS1)
LU6.2 service manager 2 AIDs	(CLS2)
LU6.2 service manager 3 AIDs	(CLS3)
Remote schedule PURGE AIDs	(CRSQ)
Resource manager resync AIDs	(CRSY)
Autoinstall terminal delete AIDs	(CATD)
Restart terminal delete AIDs	(CATR)

When a canceled SCHEDULE request is found to have a precursor in a remote CICS system—that is, the AID was originally scheduled in a remote system— then that remote AID is also canceled. The cancelation of the remote AID is performed asynchronously.

Note that CICS returns a NORMAL response even if the connection name is not defined to CICS. In this event, message DFHTF0101 indicates that no AIDs have been canceled.

FORCECANCEL
specifies that the CICS system automatic initiator descriptors (AIDs) that are queuing for the specified connection, and which cannot be removed by a simple CANCEL command, are to be forcibly removed. See Table 19 for a list of those system AIDS that require FORCECANCEL to be removed. The FORCECANCEL option can lead to unpredictable results and should be used only in exceptional circumstances.

Note that CICS returns a NORMAL response even if the connection name is not defined to CICS. In this event, message DFHTF0101 indicates that no AIDs have been canceled.

Note: FORCECANCEL does not remove transient data AIDs with an associated triggered task. These aids can be removed by purging the associated task.

SET CONNECTION conditions

Condition	***RESP2***	***Meaning***
INVREQ	16	The resource whose name was specified by CONNECTION(name) is an indirect link.
INVREQ	22	Other SET parameters were included with the CANCEL or FORCECANCEL option. Default action: terminate the task abnormally.
INVREQ	23	The resource whose name was specified by CONNECTION(name) is the local TCT system entry (TCTSE).
NORMAL	24	AIDs are successfully canceled.

End of General-use programming interface

Changes to CICS-supplied transactions

The CEMT transaction is modified to enable AIDs representing SCHEDULE or ALLOCATE requests waiting for a named terminal or connection to be canceled.

CEMT SET TERMINAL and SET CONNECTION commands

The cancel option is added to the CEMT SET TERMINAL and CEMT SET CONNECTION command:

CANCEL

AIDs queuing for the specified connection are canceled.

AIDs representing scheduled and allocated requests waiting in the local CICS system for the specified connection are canceled. However, TD AIDs with an associated triggered task already started are not canceled. Message DFHTF0100 is written to CSMT to indicate how many AIDs have been deleted for the terminal and how many remain.

See Table 19 on page 242 for a list of those system AIDS that require FORCECANCEL to be removed.

Changes to global user exits

Product-sensitive programming interface

A new global user exit point, XALCAID, is introduced. The exit is invoked only if there is data associated with the AID.

The exit is driven when an AID with data is canceled in one of the following ways:

- By means of the CEMT transaction
- During execution of a SET TERMINAL or SET CONNECTION command
- During reinstallation of a terminal or connection.

The global user exit program for XALCAID is passed a parameter list as its input.

Exit XALCAID

Invoked whenever an AID with data is canceled.

Note: It is not possible for the exit to prevent the request from being canceled.

Exit-specific parameters	**UEPALTSD**	Address of a 4-byte field containing the symbolic identifier of the transaction which was to be started by this request.
	UEPALTRM	Address of a 4-byte field containing the identifier of the terminal or connection to which this request was directed.
	UEPALDAT	The address of an area of storage containing the data specified in the FROM option, if the AID was created by a START command that specified the FROM option. For AIDs created by START requests without a FROM option, this field contains zeros.
	UEPALLEN	Address of a fullword binary value containing the length of the FROM data. If the FROM option was not specified, the field contains zeros.
	UEPALRQD	Address of an 8-byte field containing the value of the REQID associated with the FROM data. The data was stored in a temporary storage queue with this name. This value was either specified explicitly using the REQID option on the START command, or created internally by CICS.
	UEPALQUE	Address of an 8-byte field containing the value specified in the QUEUE option on the START command, or hexadecimal zeros if QUEUE was not specified.
	UEPALRTE	Address of a 4-byte field containing the value specified in the RTERMID option on the START command, or hexadecimal zeros if RTERMID was not specified.
	UEPALRTA	Address of a 4-byte field containing the value specified in the RTRANSID option on the START command, or hexadecimal zeros if RTRANSID was not specified.

Exit XALCAID (continued)

	UEPALFMH	Address of a 1-byte field containing the value X'FF' if the data contains FMHs, as specified by the FMH option on the associated START command, and X'00' otherwise.
	UEPALSTC	Address of a 2-byte field containing the start code. This is 'SZ' for FEPI starts; otherwise it is 'SD'.
Return codes		
	UERCNORM	No other return codes are supplied. The value of the return code is not inspected.
XPI calls	No XPI calls can be used.	
API and SPI commands	No EXEC CICS commands can be used.	

End of Product-sensitive programming interface

Problem determination

There are changes to trace and CICS messages to aid problem determination.

Trace changes

There are new trace entries introduced in support of the cancel start request function. These are described in the *CICS/ESA Diagnosis Reference*.

Messages and codes

There are two new messages, DFHTF0100 and DFHTF0101:

DFHTF0100 AIDs queuing for this terminal have been canceled. This could be due to the terminal being deleted, or as the result of a SPI or CEMT SET TERMINAL (termid) CANCEL command.

DFHTF0101 AIDs queuing for the named connection have been canceled or force canceled. This could be due to connection reinstall, or as the result of an EXEC CICS or CEMT SET CONNECTION (conn_name) CANCEL|FORCECANCEL command.

For details of all the new, changed, and obsolete messages, see the *CICS/ESA Migration Guide*.

Part 4. System management improvements

This Part describes the items that are introduced to improve CICS system management. It includes the following topics:

Chapter 14. Autoinstall for programs, mapsets, and partitionsets

This chapter describes the autoinstall features new to CICS/ESA 4.1. It covers the following topics:

- Overview
- Benefits of autoinstall
- Requirements for autoinstall
- Changes to CICS externals
- Problem determination.

Overview

Before the introduction of autoinstall, all resources had to be defined individually to CICS before they could be used. With the introduction of autoinstall for terminals in CICS/OS/VS 1.7, VTAM terminals could be defined dynamically on their first usage, thereby saving on storage for terminal entries, and on time spent creating the definitions.

The autoinstall function is extended in CICS/ESA 4.1 to include user programs, mapsets, and partitionsets.

In this chapter, the term 'program autoinstall' is used to mean autoinstall for all three program types (program, mapset, and partitionset) unless otherwise specified.

If the autoinstall program function is enabled, and an implicit or explicit load request is issued for a previously undefined program, mapset, or partitionset, CICS dynamically creates a definition, and installs and catalogs it, as appropriate. An implicit or explicit load occurs when:

- CICS starts a transaction.
- An application program issues one of the following commands:

  EXEC CICS LINK
  EXEC CICS XCTL
  EXEC CICS LOAD
  EXEC CICS ENABLE (for a global user exit, or task-related user exit, program).

- An application program issues a dynamic COBOL CALL
- After an EXEC CICS HANDLE ABEND PROGRAM(...) command has been issued, a condition is raised and CICS transfers control to the named program.
- CICS calls a user-replaceable module for the first time.
- An application program issues one of the following commands:

  EXEC CICS RECEIVE or SEND MAP
  EXEC CICS SEND PARTNSET
  EXEC CICS RECEIVE PARTN

Model definitions and a user-replaceable module

The program autoinstall facility uses model definitions, together with a user-replaceable module, to create explicit definitions for programs, mapsets, and partitionsets that need to be autoinstalled. CICS calls the user-replaceable module, with a parameter list that gives the name of the appropriate model definition. On return from the user-replaceable module (depending on the return code) CICS creates a resource definition from information in the model and parameters returned by the user-replaceable module.

Note that CICS does not call the user-replaceable module for any CICS programs, mapsets, or partitionsets—that is, any objects that begin with the letters DFH.

Autoinstall processing of mapsets

Table 20 shows the differences in mapset processing between a CICS region with autoinstall for programs active and a CICS region without autoinstall for programs (for example, as in CICS/ESA 3.3, or CICS/ESA 4.1 with the system initialization parameter PGAIPGM=INACTIVE).

Table 20. Differences in mapset processing between autoinstall and non-autoinstall

Program autoinstall INACTIVE	**Program autoinstall ACTIVE**
CSD definition is required. CICS attempts to load a referenced mapset with a suffix. If this fails, CICS tries an unsuffixed version. If that is unsuccessful, abend APCT is issued.	Using autoinstall, CICS attempts to load the referenced suffixed mapset or partitionset, then the unsuffixed one. Each time, if the resource is not defined, a definition is autoinstalled. The transaction requesting the resource abends only if no version of the resource exists in the library, either suffixed or unsuffixed.

Benefits of autoinstall for programs, mapsets, and partitionsets

Program autoinstall reduces system administration, virtual storage usage, and, potentially, restart times.

Reduced system administration costs

Without autoinstall, you have to define all new programs, mapsets, and partitionsets to CICS before they can be used. Autoinstall eliminates this requirement, enabling these resources to be used without prior definition. Furthermore, the need to maintain predefined definitions also disappears, leading to a significant saving in system administration effort.

Saving in virtual storage

There is a saving in virtual storage within the CICS address space, as the definitions of autoinstalled resources do not occupy table space until they are generated.

Faster restart

There is a difference in performance between warm and emergency restarts using program autoinstall **without** cataloging, and warm and emergency restarts using autoinstall **with** cataloging. (See the *CICS/ESA Recovery and Restart Guide* for information on cataloging.)

If you are using autoinstall with cataloging, restart times are similar to those of restarting a CICS region that is not using program autoinstall. This is because, in both cases, resource definitions are reinstalled from the catalog during the restart. The definitions after the restart are those that existed before the system was terminated.

If you are using autoinstall without cataloging, CICS restart times are improved because CICS does **not** install definitions from the CICS global catalog. Instead, definitions are autoinstalled as required whenever programs, mapsets, and partitionsets are referenced following the restart.

Faster cold starts: Cold starts are faster with autoinstall for programs, because you can omit all predefined program definitions from startup group lists.

System autoinstall

Some programs are autoinstalled by the **system autoinstall** function, which does not require model definitions or the support of the user-replaceable autoinstall module. The objects in this category are the transaction list table (XLT), initialization and shutdown program list tables (PLTPI and PLTSD), and the programs defined within these tables.

Requirements for autoinstall for programs, mapsets, and partitionsets

You may require a customized version of the autoinstall user-replaceable module DFHPGADX (see “Changes to user-replaceable modules” on page 259).

If you want to log messages associated with autoinstall for programs, you must define the CSPL transient data queue. See the *CICS/ESA Resource Definition Guide* for details on how to define queues.

Changes to CICS externals

The introduction of autoinstall for programs, mapsets, and partitionsets results in a number of changes to CICS externals. These are:

- Changes to system definition
- Changes to resource definition
- Changes to the system programming interface
- Changes to CICS-supplied transactions
- Changes to user-replaceable modules.

Changes to system definition

Three new system initialization parameters are added for program autoinstall. These are shown in Table 21.

Table 21. The DFHSIT macro parameters

	DFHSIT	[TYPE={**CSECT**|DSECT}] ⋮ [,PGAICTLG={**MODIFY**|NONE|ALL}] [,PGAIEXIT={**DFHPGADX**|name}] [,PGAIPGM={**INACTIVE**|ACTIVE}] ⋮
	END	DFHSITBA

PGAICTLG={MODIFY|NONE|ALL}
Specifies whether autoinstalled program definitions should be cataloged. While CICS is running, you can set whether autoinstalled programs should be cataloged dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

MODIFY
Autoinstalled program definitions are cataloged only if the program definition is modified by a SET PROGRAM command subsequent to the autoinstall.

NONE
Autoinstalled program definitions are not cataloged. This gives a faster CICS restart (warm and emergency) compared with the MODIFY or ALL options, because CICS does not reinstall definitions from the global catalog. Definitions are autoinstalled on first reference.

ALL
Autoinstalled program definitions are written to the global catalog at the time of the autoinstall, and following any subsequent modification.

PGAIEXIT={DFHPGADX|name}
Specifies the name of the program autoinstall exit program. While CICS is running, you can set the name of the program autoinstall exit program dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

PGAIPGM={INACTIVE|ACTIVE}
Specifies the state of the program autoinstall function at initialization. While CICS is running, you can set the status of program autoinstall dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

INACTIVE
The program autoinstall function is disabled.

ACTIVE
The program autoinstall function is enabled.

Changes to resource definition

There are no changes to the syntax of resource definition commands, but the concept of program and map definitions being used as models is introduced. Also, for program definitions that you predefine in the CSD, you no longer need to specify the language of the program—CICS determines this when loading the program.

The purpose of a model definition is to provide CICS with one definition that can be used for all programs with the same properties. Any properties unique to a program, or which you do not want to default, can be specified in the program autoinstall user-replaceable module (see "Changes to user-replaceable modules" on page 259). If you specify a model name by means of the user-replaceable module, CICS uses that model. If no model is specified, CICS uses the default model appropriate to the program type (that is, program, mapset, or partitionset).

Any installed program definition can be used as a model for an autoinstalled program, or you can use the default model definitions supplied by CICS for each program type. The default model definitions are supplied as follows:

- Default program definition DFHPGAPG:

```
GROUP(DFHPGAIP)       PROGRAM(DFHPGAPG)
DESCRIPTION(Default model program definition for program autoinstall)
+                     RELOAD(NO)            RESIDENT(NO)
USAGE(NORMAL)         USELPACOPY(NO)        STATUS(ENABLED)
CEDF(YES)             DATALOCATION(BELOW)   EXECKEY(USER)
EXECUTIONSET(FULLAPI)
```

  + Although this default program resource definition does not specify a program
  + language, the CICS autoinstall routine detects the program language from the program being loaded, and sets the language field accordingly.

- Default mapset definition DFHPGAMP:

```
GROUP(DFHPGAIP)       MAPSET(DFHPGAMP)
DESCRIPTION(Default model mapset definition for program autoinstall)
RESIDENT(NO)    USAGE(NORMAL)    USELPACOPY(NO)    STATUS(ENABLED)
```

- Default partitionset definition DFHPGAPT:

```
GROUP(DFHPGAIP)     PARTITIONSET(DFHPGAPT)
DESCRIPTION(Default model partitionset defn for program autoinstall)
RESIDENT(NO)     USAGE(NORMAL)     USELPACOPY(NO)    STATUS(ENABLED)
```

All the above default resource definitions to support autoinstall for programs are provided in the CSD group DFHPGAIP, which is automatically included in the CICS startup list, DFHLIST. When creating your own startup group list, you should ensure that the DFHPGAIP group is included.

If you want to create your own definitions, you must use RDO or the DFHCSDUP utility (see the *CICS/ESA Resource Definition Guide* for information on how to do this).

Changes for system-autoinstalled programs

Some programs no longer need resource definitions, nor do they need the support of autoinstall model definitions or the user-replaceable autoinstall module. These are programs that are in the system autoinstall category. Programs in this category are system autoinstalled automatically (if they are not already defined in the CSD). This function is independent of the program autoinstall function.

Programs that are system autoinstalled are assigned the following program attributes:

```
Status:          ENABLED    Execution set:      FULLAPI
Execution key:   CICS       Program attribute:  RESUSEABLE
CEDF status:     NOCEDF     Program usage:      APPLICATION
Language:        ASSEMBLER  Share status:       TYPE_ANY
Data Location:   BELOW
```

Note: The language defined at system autoinstall is replaced by the actual language on first use of the program.

The programs that are covered by the system autoinstall function are as follows:

Program list tables (PLTPI and PLTSD)

You do not need to define program definitions in the CSD for DFHPLT*xx* tables defined on the PLTPI and PLTSD system initialization parameters. The PLT tables are system autoinstalled.

Note: Whether the programs that are defined in the PLTs are system autoinstalled, or autoinstalled by the program autoinstall function, depends on the phase of the PLT they are in, and also on the startup or shutdown phase.

Transaction list tables (XLTs)

You do not need to define a PROGRAM definition in the CSD for the DFHXLT*xx* table specified on the XLT system initialization parameter. The XLT is system autoinstalled.

Autoinstall for programs defined in a list table: PLTPI programs that are defined *before* the table delimiter DFHDELIM, and PLTSD programs that are defined *after* DFHDELIM are in the system autoinstall category. They do not need program resource definitions in the CSD, and are system autoinstalled independent of the program autoinstall function.

PLTPI programs that are defined *after* the table delimiter DFHDELIM, and PLTSD programs that are defined *before* DFHDELIM are eligible for autoinstall. Program autoinstall must be active, otherwise they must be defined by program resource definitions in the CSD.

Changes to the system programming interface

General-use programming interface

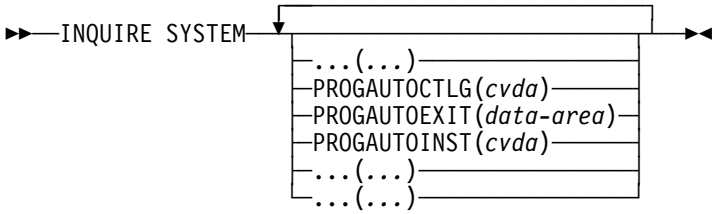
Three new options are added to the EXEC CICS INQUIRE and EXEC CICS SET SYSTEM commands for program autoinstall.

EXEC CICS INQUIRE SYSTEM command

Function

Retrieve information about system definitions.

Syntax



Conditions:
NOTAUTH

INQUIRE SYSTEM options

PROGAUTOCTLG(cvda)
returns a CVDA value indicating whether autoinstalled program definitions are to be cataloged. CVDA values are:

CTLGALL
All autoinstalled program definitions are to be cataloged and restored on a warm or emergency start.

CTLGMODIFY
Autoinstalled program definitions are to be cataloged only if they are modified (for example, by a SET PROGRAM command), so that the modified definitions are restored on a warm or emergency restart.

CTLGNONE
No autoinstalled program definitions are to be cataloged. They are autoinstalled again after a warm or emergency start.

PROGAUTOEXIT(data-area)
returns an 8-character name of the user-provided program that is called by the program autoinstall code to select or modify a model definition.

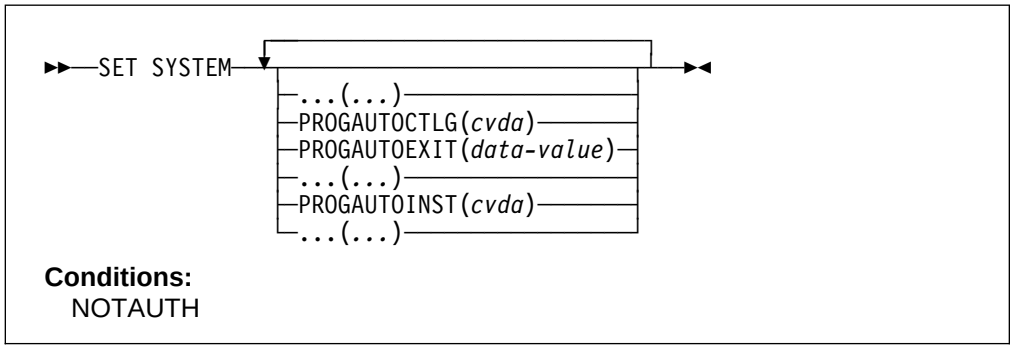
PROGAUTOINST(cvda)
returns a CVDA value indicating whether autoinstall for programs is active or inactive. CVDA values are:

AUTOACTIVE
Autoinstall for programs is active. On first use, if a program, mapset, or partitionset is not defined, CICS creates the definition dynamically.

AUTOINACTIVE
Autoinstall is not active. If a program is not defined, a PGMIDERR or transaction abend occurs when it is referenced.

EXEC CICS SET SYSTEM command

Function
Specify new system data for a CICS region.

Syntax


Conditions:
NOTAUTH

SET SYSTEM options
PROGAUTOCTLG(cvda)
specifies whether autoinstalled program definitions are to be cataloged. CVDA values are:

CTLGALL
All autoinstalled program definitions are to be cataloged and restored on a warm or emergency start.

CTLGMODIFY
Autoinstalled program definitions are to be cataloged only if they are modified (for example, by a SET PROGRAM command), so that the modified definitions are restored on a warm or emergency restart.

CTLGNONE
No autoinstalled program definitions are to be cataloged. They are autoinstalled again after a warm or emergency start.

PROGAUTOEXIT(data-value)
specifies the name of the user-provided program that is to be called by the CICS autoinstall program to select or modify a model definition.

Note that your autoinstall program cannot itself be autoinstalled, nor can any program it references. You must define a program resource definition for your autoinstall program, and any other program it references, in the CSD. You must also ensure these definitions are installed in the CICS region. If you specify an invalid name for the autoinstall program, CICS disables the program, thus disabling the program autoinstall function.

PROGAUTOINST(cvda)
specifies whether autoinstall for programs is to be active or inactive. CVDA values are:

AUTOACTIVE
Autoinstall for programs is active. On first use, if a program, mapset, or partitionset is not defined, the definition is created dynamically.

AUTOINACTIVE
Autoinstall is not active. If a program is not defined, a PGMIDERR or transaction abend occurs when it is referenced.

End of General-use programming interface

Changes to CICS-supplied transactions

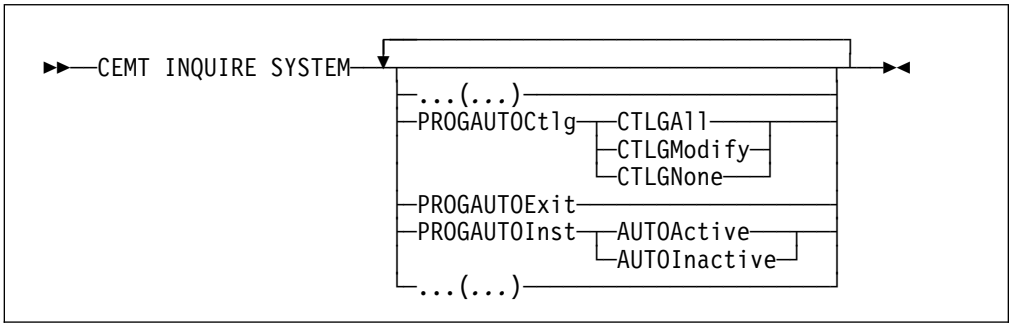
Three new options are added to the CEMT INQUIRE SYSTEM and CEMT SET SYSTEM command for program autoinstall.

CEMT INQUIRE SYSTEM command

Function

Retrieve information about a named system definition.

Syntax



INQUIRE SYSTEM options

PROGAUTOCtlg(value)
displays whether autoinstalled program definitions are to be cataloged. The values displayed are:

CTLGALL
All autoinstalled program definitions are to be cataloged and restored on a warm or emergency start.

CTLGMODIFY
Autoinstalled program definitions are to be cataloged only if they are modified (for example, by a SET PROGRAM command), so that the modified definitions are restored on a warm or emergency restart.

CTLGNONE
No autoinstalled program definitions are to be cataloged. They are autoinstalled again after a warm or emergency start.

PROGAUTOExit(value)
displays the name of the user-provided program that is called by the program autoinstall code to select or modify a model definition.

PROGAUTOInst(value)
displays whether autoinstall for programs is active or inactive. The values are:

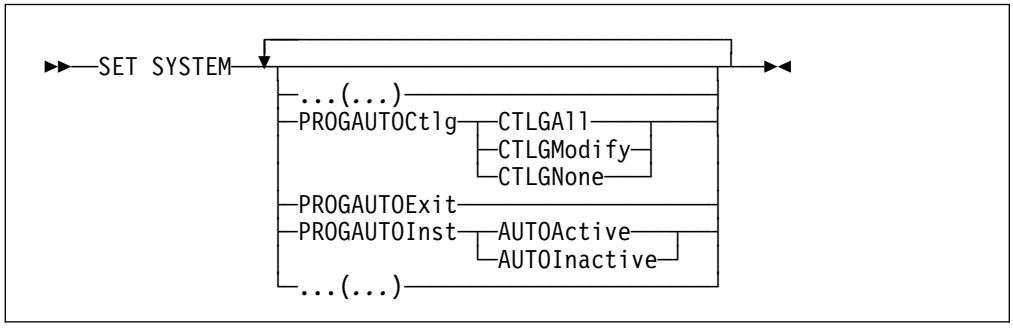
AUTOACTIVE
Autoinstall for programs is active. On first use, if a program, mapset, or partitionset is not defined, the definition is created dynamically.

AUTOINACTIVE
Autoinstall is not active. If a program is not defined, a PGMIDERR or transaction abend occurs when it is referenced.

CEMT SET SYSTEM command

Function
Change information for a named system definition.

Syntax



SET SYSTEM options
The new options are:

PROGAUTOCtlg(value)
specifies whether autoinstalled program definitions are to be cataloged. The values are you can specify are:

CTLGALL
All autoinstalled program definitions are to be cataloged and restored on a warm or emergency start.

CTLGMODIFY
Autoinstalled program definitions are to be cataloged only if they are modified (for example, by a SET PROGRAM command), so that the modified definitions are restored on a warm or emergency restart.

CTLGNONE
No autoinstalled program definitions are to be cataloged. They are autoinstalled again after a warm or emergency start.

PROGAUTOExit(value)
specifies the name of the user-provided program that is to be called by the CICS autoinstall program to select or modify a model definition.

Note that your autoinstall program cannot itself be autoinstalled, nor can any program it references. You must define a program resource definition for your autoinstall program, and any other program it references, in the CSD. You must also ensure these definitions are installed in the CICS region. If you

specify an invalid name for the autoinstall program, CICS disables the program, thus disabling the program autoinstall function.

PROGAUTOInst(value)
specifies whether autoinstall for programs is to be active or inactive. The values are:

AUTOACTIVE
Autoinstall for programs is active. On first use, if a program, mapset, or partitionset is not defined, the definition is created dynamically.

AUTOINACTIVE
Autoinstall is not active. If a program is not defined, a PGMIDERR or transaction abend occurs when it is referenced.

Changes to user-replaceable modules

Product-sensitive programming interface

There are no changes to existing user-replaceable modules, but a new user-replaceable module is introduced for program autoinstall.

The purpose of the new autoinstall user-replaceable module is to provide CICS with the extra information it needs to complete an autoinstall request, such as the autoinstall model name.

You can write your autoinstall program in any language supported by CICS, with full access to the CICS application and system programming interfaces.

Note: Your autoinstall program cannot itself be autoinstalled, nor can any program it references. You must define a program resource definition in the CSD for your autoinstall program and for any other programs it references. You must also ensure these definitions are installed in the CICS region during startup by including the group containing the definitions in your startup group list. If you specify an invalid name for the autoinstall program, CICS disables the program, thus disabling the program autoinstall function.

The following program resource definitions are supplied by CICS for the autoinstall control program; the default is the assembler version, DFHPGADX. If these definitions are not suitable for your use, you can create your own, using RDO or the DFHCSDUP utility.

- Default autoinstall control program definition for DFHPGADX. This defines the assembler version, and its status is set to ENABLED:

```
GROUP(DFHPGAIP)       PROGRAM(DFHPGADX)
DESCRIPTION(Assembler definition for program autoinstall exit)
LANGUAGE(ASSEMBLER)   EXECKEY(CICS)         EXECUTIONSET(FULLAPI)
RELOAD(NO)            RESIDENT(NO)          USAGE(NORMAL)
STATUS(ENABLED)       CEDF(NO)              DATALOCATION(ANY)
```

- Autoinstall control program definition for DFHPGAOX. This defines the CICS-supplied COBOL version, and its status is set to DISABLED:

```
GROUP(DFHPGAIP)       PROGRAM(DFHPGAOX)
DESCRIPTION(COBOL definition for program autoinstall exit)
LANGUAGE(COBOL)       EXECKEY(CICS)         EXECUTIONSET(FULLAPI)
RELOAD(NO)            RESIDENT(NO)          USAGE(NORMAL)
STATUS(DISABLED)      CEDF(NO)              DATALOCATION(ANY)
```

- Autoinstall control program definition for DFHPGAHX. This defines the CICS-supplied C/370 version, and its status is set to DISABLED:

```
GROUP(DFHPGAIP)       PROGRAM(DFHPGAHX)
DESCRIPTION(C definition for program autoinstall exit)
LANGUAGE(C)           EXECKEY(CICS)         EXECUTIONSET(FULLAPI)
RELOAD(NO)            RESIDENT(NO)          USAGE(NORMAL)
STATUS(DISABLED)      CEDF(NO)              DATALOCATION(ANY)
```

- Autoinstall control program definition for DFHPGALX. This defines the CICS-supplied PL/I version, and its status is set to DISABLED:

```
GROUP(DFHPGAIP)       PROGRAM(DFHPGALX)
DESCRIPTION(PL/I definition for program autoinstall exit)
LANGUAGE(PLI)         EXECKEY(CICS)         EXECUTIONSET(FULLAPI)
RELOAD(NO)            RESIDENT(NO)          USAGE(NORMAL)
STATUS(DISABLED)      CEDF(NO)              DATALOCATION(ANY)
```

Supported languages for the program autoinstall control program

You can write the user-replaceable module for program autoinstall in any one of the four supported languages—assembler, COBOL, PL/I, and C language. CICS provides a version of the default program in all four languages:

DFHPGADX Assembler
DFHPGAOX COBOL
DFHPGAHX C
DFHPGALX PL/I

The source for these versions of the default program is supplied in CICS410.SDFHSAMP, but only the assembler version is supplied in executable form, in CICS410.SDFHLOAD.

General rules for writing your autoinstall control program

You should note the following when writing your own control program for program autoinstall.

- **Input** The first two fields of the parameter list are input-only fields and should not be altered by your program.
- **Output** The remaining fields on the parameter list are input/output or output only fields, which you can use to specify attributes that override those of the model definition.
- Some of the output fields in the parameter list are not applicable to mapsets or partitionsets. CICS ignores any parameters you specify that are not applicable to the type of object being installed.
- Any attributes you return to CICS in the parameter list are used to modify the model definition, and CICS installs the modified definition. Once installed, the definition can be modified normally using the EXEC CICS SET PROGRAM or CEMT SET PROGRAM commands.
- If you modify your control program, you can make the new version available by using the EXEC CICS SET PROGRAM NEWCOPY or CEMT SET PROGRAM NEWCOPY command.
- You can discard definitions after they have been installed; they are reinstalled when next referenced.

- You must ensure that the parameters you return to CICS are valid, and consistent with other system attributes in your CICS region. For example:
  - Do not return PGAC_LPA_YES on the PGAC_USE_LPA_COPY parameter if CICS is running with the system initialization parameter LPA=NO.
  - Do not return PGAC_USER_KEY (which is the default) on the PGAC_EXECUTION_KEY parameter if the task for which your control program is called is running with CICS-key task-lifetime storage.

    You can determine the storage key for the task by testing the TASKDATAKEY option in its transaction definition by means of the following EXEC CICS commands:

    ```
    EXEC CICS ADDRESS EIB
    EXEC CICS INQUIRE TRANSACTION(eibtrans) TASKDATAKEY(...)
    ```

The autoinstall parameter list

CICS passes the information you need to tailor the user-replaceable module as a parameter list in a COMMAREA. The parameter list is supplied in a form suitable for each of the supported languages, in the library appropriate to the language:

DFHPGACD Assembler copybook, in CICS410.SDFHMAC
DFHPGACO COBOL copybook, in CICS410.SDFHCOB
DFHPGACH C copybook, in CICS410.SDFHC370
DFHPGACL PL/I copybook, in CICS410.SDFHPL1

The assembler form of the parameter list is as follows:

PGAC_PROGRAM
passes the 8-byte name of the object to be autoinstalled. This is an input-only field, which your user-replaceable module must not alter.

PGAC_MODULE_TYPE
passes a 1-byte indicator for the type of object to be installed—program, mapset, or partitionset.

The equated values for the type field are:

PGAC_TYPE_PROGRAM
The object to be installed is a program.

PGAC_TYPE_MAPSET
The object to be installed is a mapset.

PGAC_TYPE_PARTITIONSET
The object to be installed is a partitionset.

This is an input-only field, which your user-replaceable module must not alter.

PGAC_MODEL_NAME
specifies the 8-byte autoinstall model name, which, on invocation, is set to the default model name for the type of object:

DFHPGAPG For a program.

DFHPGAMP For a mapset.

DFHPGAPT For a partitionset.

PGAC_LANGUAGE
specifies, in a 1-byte field, the language for the program—assembler, COBOL, C, Language Environment/370, or PL/I.

The autoinstall routine determines the program language from the program being loaded, and sets this language field accordingly.

The equated values for the language field are:

PGAC_ASSEMBLER
The program is an assembler program.

PGAC_COBOL
The program is a COBOL program.

PGAC_C370 The program is a C program.

PGAC_LE370 The program is Language Environment/370 program.

PGAC_PLI The program is a PL/I program.

PGAC_CEDF_STATUS
specifies, in a 1-byte field, the execution diagnostic facility (EDF) status for the program to be autoinstalled.

The equated values for the EDF status are:

PGAC_CEDF_YES The EDF facility can be used for this program.

PGAC_CEDF_NO The EDF facility cannot be used for this program.

PGAC_DATA_LOCATION
specifies, in a 1-byte field, the data location for task-lifetime storage.

The equated values for task-lifetime storage are:

PGAC_LOCATION_BELOW
Task-lifetime storage must be located below 16MB.

PGAC_LOCATION_ANY
Task-lifetime storage can be below or above 16MB.

PGAC_EXECUTION_KEY
specifies, in a 1-byte field, the execution key for the program.

The equated values for the execution key are:

PGAC_CICS_KEY The program is to execute in CICS key.
PGAC_USER_KEY The program is to execute in user key.

PGAC_LOAD_ATTRIBUTE
specifies, in a 1-byte field, the load attributes for the object.

The equated values for load attributes are:

PGAC_RELOAD
specifies that CICS is to load a fresh copy of the object for each request.

PGAC_RESIDENT
specifies that CICS is to make the object permanently resident.

PGAC_TRANSIENT
specifies that the storage for this object is to be released whenever the use count reaches zero.

PGAC_REUSABLE
specifies that CICS can use any copy of the object currently in storage.

PGAC_USE_LPA_COPY
specifies, in a 1-byte field, whether CICS is to use an LPA-resident copy of the program.

The equated values for the LPA status are:

PGAC_LPA_YES
specifies that CICS is to use a copy form the LPA.

PGAC_LPA_NO
specifies that CICS is to load a private copy from its own DFHRPL library concatenation.

PGAC_EXECUTION_SET
specifies, in a 1-byte field, whether the program is restricted to the distributed program link (DPL) subset of the API, or whether it can use the full API.

The equated values for the API execution set are:

PGAC_DPLSUBSET
The program is to be restricted to the DPL subset.

PGAC_FULLAPI
The program can use the full API.

PGAC_REMOTE_SYSID
specifies, in a 4-byte field, the name of the remote system where the program is to execute if the program invocation is be treated as a distributed program link (DPL) request. CICS function ships any request for this program to the specified remote CICS.

PGAC_REMOTE_PROGID
specifies, in an 8-byte field, the name by which the program is known in the remote CICS region. For a remote program, the remote name defaults to the local name if you set this field to blank.

PGAC_REMOTE_TRANSID
specifies, in a 4-byte field, the name of the CICS mirror transaction under which a remote program is to run. By default, this is set to the name of the CICS mirror transaction, CSMI.

PGAC_RETURN_CODE
specifies, in a 1-byte field, the autoinstall program return code to CICS.

The equated values for the return code are:

PGAC_RETURN_OK
Install the program definition using the values returned in the COMMAREA parameter list.

PGAC_RETURN_DONT_DEFINE_PROGRAM
Do not define the program.

Note: Any parameters that are not applicable to programs are undefined.

Problem determination

There are new messages issued by the program manager domain relating to program autoinstall. All program autoinstall messages are written to the CSPL transient data queue.

All new and changed messages in CICS/ESA 4.1 are listed in the *CICS/ESA Migration Guide*.

Testing and debugging your program autoinstall control program

You can use the CICS execution diagnostic facility (EDF) to help you test your autoinstall control program. However, EDF is inhibited for programs with names that begin with the letters DFH; so to use EDF you must name your program something other than one of the default names.

End of Product-sensitive programming interface

Chapter 15. Autoinstall for APPC connections

This chapter describes the autoinstall facility introduced in CICS/ESA 4.1 for APPC connections. It covers the following topics:

- Overview
- Benefits of autoinstall for APPC connections
- Requirements for autoinstall for APPC connections
- Changes to CICS externals
- Problem determination.

Overview

Before the introduction of autoinstall, all resources had to be defined individually to CICS before they could be used. With the introduction of autoinstall for terminals in CICS/OS/VS 1.7, VTAM terminals could be defined dynamically on their first usage, thereby saving on storage for terminal entries, and on time spent creating the definitions.

Autoinstall is extended in CICS/ESA 4.1 to include support for

- APPC parallel sessions
- APPC single sessions.

In each of these cases the additional support is for BIND requests from primary nodes.

Autoinstall for APPC single sessions initiated by a CINIT request

CICS/ESA 4.1 continues to support autoinstall for APPC single sessions initiated by a CINIT request, as in earlier releases. This works in the same way as autoinstall for terminals, in that you supply a model TERMINAL definition and its associated TYPETERM definition.

APPC parallel sessions and single sessions initiated by a BIND

If autoinstall is enabled, and an APPC BIND request is received for an APPC service manager (SNASVCMG) session for which CICS does not have a matching CONNECTION definition, CICS creates a new connection and installs it automatically.

Like autoinstall for terminals, autoinstall for connections uses the concept of a model definition on which to base new definitions. Unlike autoinstall for terminals, however, the autoinstall model definitions for connections are not defined explicitly as models. You can use any previously-installed connection definition as a model for the new definition. However, you are recommended to define connection and sessions definitions that are used only for autoinstall purposes, and to install these only when you are using the APPC autoinstall function.

In order for autoinstall to work, you must have a model for each kind of connection you want to be autoinstalled. You also need a terminal autoinstall user-replaceable module that is capable of handling APPC install requests (see “Changes to user-replaceable modules” on page 269 for details).

Recovery and restart

Autoinstalled connections for parallel and single sessions originating from a BIND are not cataloged. This means they are not recovered at an emergency restart or a warm restart.

Autoinstall models for APPC connections

The purpose of a model is to provide CICS with a definition that can be used as a template for all connections with the same properties. The terminal autoinstall user-replaceable module (see “Changes to user-replaceable modules” on page 269) selects an appropriate model for each new connection, depending on the information it receives from VTAM.

For APPC autoinstall, a model consists of a CONNECTION definition and its associated SESSIONS definition(s). You should have suitable model definitions installed to act as templates for each different set of session properties that you need.

Because the models used for APPC autoinstall are not defined explicitly as models, they are not displayed when you inquire on autoinstall models (using either the CEMT INQUIRE AUTINSTMODEL or EXEC CICS INQUIRE AUTINSTMODEL).

Although you can use any installed connection definition as a model, for performance reasons you are recommended to install definitions that are used only as models (or templates) for autoinstall. The definition used as a model is locked while CICS is taking a copy to autoinstall an APPC connection. If you have a large number of sessions being autoinstalled at the same time, the delay could affect other users trying to get access to the table entry.

Benefits of autoinstall for APPC connections

Autoinstall support provides the most benefit if you have a large number of APPC parallel session devices with identical characteristics. For example, if you have 1000 PS/2s, all with the same characteristics, you can set up one model to autoinstall all of them. If 500 of your PS/2s have one set of characteristics, and 500 have a different set, you can set up two models to autoinstall all the PS/2s in each set.

Restart of any kind should be noticeably faster, especially when large numbers of terminals are involved.

Savings can also be made on systems management overheads, and on storage, as autoinstalled resources do not occupy space until they are required.

Requirements for autoinstall for APPC connections

You require a terminal autoinstall program that CICS can call to autoinstall the necessary connection definitions. You can use the CICS-supplied version of the terminal autoinstall program, DFHZATDY (see “Changes to user-replaceable modules” on page 269), but it is probable that you will need to customize this to your own requirements.

Changes to CICS externals

The following changes have been made to CICS externals:

- Changes to system definition
- Changes to resource definition
- Changes to user-replaceable modules.

Changes to system definition

There are no new or obsolete system initialization parameters in CICS/ESA 4.1 for autoinstall of APPC connections, but the function of two of the existing terminal autoinstall system initialization parameters is extended to APPC connection autoinstall. The affected parameters are AIEXIT and AIQMAX, which are described as follows:

AIEXIT={DFHZATDX|name}

Code this parameter with the name of the autoinstall user-replaceable module that you want CICS to use when autoinstalling VTAM terminals and APPC connections and sessions. Autoinstall is the process of installing resource definitions automatically, using VTAM logon or BIND data, model definitions, and an autoinstall program.

Note: You specify only one user-replaceable program to handle autoinstall for both terminals and APPC connections.

For more information about autoinstall, see the *CICS/ESA Resource Definition Guide*.

DFHZATDX

The CICS-supplied autoinstall user program, DFHZATDX, is the default. This program handles terminals only. If you want to autoinstall both terminals and APPC connections, either specify the CICS-supplied user program, DFHZATDY, or a customized version of this program.

name

The name of your own customized autoinstall user-replaceable module. For more information about writing your own autoinstall program, which may perform additional processing, see the *CICS/ESA Customization Guide*.

AIQMAX={100|number}

Code this parameter with the maximum number of VTAM terminals and APPC connections that can be queued concurrently for autoinstall.

number

A number in the range 0 through 999. The default is 100.

A zero value disables the autoinstall function.

You should specify a number that is large enough to allow for both APPC connections and terminals.

Note: This value does not limit the total number of terminals that can be autoinstalled. If you have a large number of terminals autoinstalled, shutdown can fail due to the MXT system initialization parameter being reached or CICS becoming short on storage. For more information about preventing this possible cause of shutdown failure, see the *CICS/ESA Performance Guide*.

The other autoinstall system initialization parameters, AIRDELAY and AILDELAY, have no relevance to APPC connections, and are meaningful for terminal autoinstall only.

Changes to resource definition

A new program definition, three new connection definitions, and three new session definitions are introduced for the sample control program for connection autoinstall. They are supplied in the CICS-supplied CSD group, DFHAI62, when you initialize, or upgrade, your CSD.

The connection and sessions definitions are provided for use with the CICS-supplied autoinstall program, DFHZATDY, and are shown in Figure 19:

```
DEFINE CONNECTION(CBPS)  GROUP(DFHAI62)  NETNAME(TMPLATE1)
       DESCRIPTION(Connection for parallel sessions initiated by BIND)
       ACCESSMETHOD(VTAM)  PROTOCOL(APPC)  SINGLESESS(NO)

DEFINE SESSION(CBPS)  GROUP(DFHAI62)  CONNECTION(CBPS)
       DESCRIPTION(For parallel sessions initiated by BIND)
        PROTOCOL(APPC)  MAXIMUM(10,5)

DEFINE CONNECTION(CBSS)  GROUP(DFHAI62)  NETNAME(TMPLATE2)
       DESCRIPTION(Connection for single session initiated by BIND)
       ACCESSMETHOD(VTAM)  PROTOCOL(APPC)  SINGLESESS(YES)

DEFINE SESSION(CBSS)  GROUP(DFHAI62)  CONNECTION(CBSS)
       DESCRIPTION(For single session initiated by BIND)
        PROTOCOL(APPC)  MAXIMUM(1,0)

DEFINE CONNECTION(CCPS)  GROUP(DFHAI62)  NETNAME(TMPLATE3)
       DESCRIPTION(For parallel sessions initiated by CINIT)
       ACCESSMETHOD(VTAM)  PROTOCOL(APPC)  SINGLESESS(NO)

DEFINE SESSION(CCPS)  GROUP(DFHAI62)  CONNECTION(CCPS)
       DESCRIPTION(For parallel sessions initiated by CINIT)
        PROTOCOL(APPC)  MAXIMUM(10,5)
```

Figure 19. Connection and session definitions for use with autoinstall for APPC connections

Note: The CICS-supplied group DFHAI62 also contains definitions for parallel sessions received from a secondary node via a CINIT request, but these requests cannot be received by CICS/ESA 4.1.

Also, the SESSIONS definitions do not specify the MODENAME parameter. If you plan to use these definitions, and a blank modename is not appropriate for your use, edit these definitions to provide a valid modename.

“Changes to user-replaceable modules” on page 269 contains details of the autoinstall control program definition that uses the above definitions.

Changes to user-replaceable modules

Product-sensitive programming interface

CICS provides a new terminal autoinstall user-replaceable module, DFHZATDY, which you can use instead of DFHZATDX as the basis for developing your own autoinstall control program. DFHZATDY provides the same function for terminal autoinstall as the DFHZATDX module of earlier releases, but in addition it provides function to autoinstall APPC connections for single and parallel sessions.

Note: CICS/ESA 4.1 provides versions of the terminals-only autoinstall control program in all the supported languages—DFHZATDX (assembler); DFHZCTDX (COBOL); DFHZPTDX (PL/I); and DFHZDTDX (C), as in earlier releases, but only the assembler version is modified to include basic support for the APPC autoinstall function.

The CICS-supplied resource definition for the terminal and connection autoinstall control program, DFHZATDY, is included in the CSD group, DFHAI62, as follows:

```
DEFINE PROGRAM(DFHZATDY)
DESCRIPTION(Assembler version of the terminal autoinstall program)
GROUP(DFHAI62)
LANGUAGE(ASSEMBLER) RELOAD(NO)          RESIDENT(NO)
USAGE(NORMAL)       STATUS(ENABLED)     CEDF(NO)
DATALOCATION(ANY)   EXECKEY(CICS)       EXECUTIONSET(FULLAPI)
```

The source for this program is supplied in the CICS410.SDFHSAMP library.

The purpose of the terminal autoinstall control program is to provide CICS with any extra information it needs to complete an autoinstall request. For APPC connections, the terminal autoinstall control program must provide a connection name for use as the system identifier SYSID for the new definition being installed.

When CICS receives an APPC BIND request, it contains the partner's NETNAME which CICS passes to your terminal autoinstall program.

The CICS-supplied terminal autoinstall control program uses information from the BIND, which is passed in the COMMAREA, to select the most appropriate model on which to base a new connection.

Your terminal autoinstall control program needs to know the NETNAME or the CONNECTION name of all the available model connection definitions, in order to return the name of the most suitable one. If it attempts to use an unsuitable definition as a template, CICS issues message DFHZC6922, explaining why the template is unusable. See “New messages” on page 272 for details.

If the template is usable, CICS makes a copy of it and attempts to install the new CONNECTION definition. If the install is not successful, CICS issues message DFHZC6921; see “New messages” on page 272 for details.

The terminal autoinstall control program interface

The terminal autoinstall control program of earlier CICS releases is invoked for an installation or deletion request against an autoinstalled terminal or APPC single session initiated by a CINIT.

With the introduction of autoinstall for APPC connections, the autoinstall control program is now also invoked at installation for:

- APPC parallel sessions initiated by a BIND
- APPC single sessions initiated by a BIND.

On each invocation of the autoinstall control program, CICS passes a parameter list to the control program in a communication area (COMMAREA). This describes the function being performed (in this case INSTALL), and provides data relevant to the install function.

Note: There is no delete function for APPC connections equivalent to the delete function for autoinstalled terminals.

The COMMAREA at INSTALL for APPC parallel sessions

The layout of the COMMAREA, which is addressed by DFHEICAP, is shown in Figure 20. This COMMAREA is mapped by the DSECT for the assembler version of DFHZATDY, which is supplied in CICS410.SDFHMAC.

```
*-------------------------------------------------------------------*
* APPC Install parameter list - Functions 2, 3, and 4               *
*-------------------------------------------------------------------*
INSTALL_APPC_COMMAREA       DSECT       Install Parameter List
*
INSTALL_APPC_STANDARD       DS  F       Standard field
                            ORG INSTALL_APPC_STANDARD
INSTALL_APPC_EXIT_FUNCTION  DS  XL1     Install request type
INSTALL_APPC_PS_CINIT       EQU X'F2'   Install PS via CINIT
INSTALL_APPC_PS_BIND        EQU X'F3'   Install PS via BIND
INSTALL_APPC_SS_BIND        EQU X'F4'   Install SS via BIND
INSTALL_APPC_EXIT_COMPONENT DS  CL2     Component ID 'ZC'
                            DS  XL1     Reserved
*
                            ORG ,
INSTALL_APPC_NETNAME_PTR    DS  A       -> NETNAME          Input
INSTALL_APPC_CINIT_PTR      DS  0A      -> CINIT_RU         Input
INSTALL_APPC_BIND_PTR       DS  A       -> BIND             Input
INSTALL_APPC_SELECTED_PTR   DS  A       -> return fields    Output
INSTALL_APPC_SYNCLEVEL_PTR  DS  A       -> Sync level       Input
*
INSTALL_APPC_TEMPLATE_NETNAME_PTR DS A  -> template NETNAME Output
INSTALL_APPC_TEMPLATE_SYSID_PTR DS A    -> template SYSID   Output
INSTALL_APPC_SYSID_PTR      DS A        -> new SYSID        Output
*
TEMPLATE_NETNAME            DS CL8      user puts netname here
TEMPLATE_SYSID              DS CL4      user puts sysid   here
SYSID                       DS CL4      user puts new name here
SYNCLEVEL                   DS XL2      synclevel of new connection
```

Figure 20. Autoinstall control program's communications area at INSTALL. For APPC connections initiated by BIND requests.

INSTALL_APPC_STANDARD header
A fullword input field comprising the following information:

INSTALL_APPC_EXIT_FUNCTION
A 1-byte field that defines the install request type. The equated values are:

INSTALL_APPC_PS_CINIT
X'F2' represents an install request for an APPC parallel-session connection from a secondary node via a CINIT request.

Note: These requests cannot be received by CICS/ESA 4.1.

INSTALL_APPC_PS_BIND
X'F3' represents an install request for an APPC parallel-session connection via a BIND.

INSTALL_APPC_SS_BIND
X'F4' represents an install request for an APPC single-session connection via a BIND.

Note: The values X'F0' and X'F1' represent, respectively, install and delete requests for terminals (including APPC single-session devices).

INSTALL_APPC_EXIT_COMPONENT
A 2-byte component code, which is set to 'ZC'.

INSTALL_APPC_NETNAME_PTR
A fullword pointer to a 2-byte length field, followed by the NETNAME to be installed (input field).

INSTALL_APPC_CINIT_PTR
A fullword pointer to an input field containing the incoming CINIT, if the incoming session is a secondary.

Note: Not applicable to CICS/ESA 4.1.

INSTALL_APPC_BIND_PTR
A fullword pointer to an input field containing the incoming BIND.

INSTALL_APPC_SELECTED_PTR
A fullword pointer to the return fields. These are in the same format as those for autoinstall of terminals.

Note that for APPC autoinstall (functions X'F3' and X'F4') only the return code is used. You return other information for APPC in other fields defined in the communications area.

INSTALL_APPC_SYNCLEVEL_PTR
A fullword pointer to a 2-byte input field specifying the syncpoint level for the connection, which is extracted from the BIND. The possible values are:

X'0000' Synclevel 0
X'0001' Synclevel 1
X'0002' Synclevel 2.

INSTALL_APPC_TEMPLATE_NETNAME_PTR
A fullword pointer to an 8-byte output area that your control program can use to specify the NETNAME of the template. If the name is less than 8 bytes, it must be padded with trailing blanks. If, alternatively, you specify a CONNECTION name (in the SYSID field), the 8-byte area should be filled with zeros.

INSTALL_APPC_TEMPLATE_SYSID_PTR
A fullword pointer to a 4-byte output area that your control program can use to specify the SYSID (connection name) of the template. If the name is less than 4 bytes, it must be padded with trailing blanks. If, alternatively, you specify a NETNAME, the 4-byte area should be filled with zeros.

INSTALL_APPC_SYSID_PTR
A fullword pointer to a 4-byte output area in which your program must put the SYSID for the new autoinstalled connection. The name you supply must be unique. You can use the same or similar logic to create it that you use for creating a terminal ID. If the name is less than 4 bytes, it must be padded with trailing blanks.

If you are using recoverable resources, the SYSID chosen for a connection after a restart must be the same as that chosen in the previous CICS run.

End of Product-sensitive programming interface

Problem determination

There are some new and changed messages to help with problem determination with the APPC autoinstall function.

New messages

The following messages have been added.

- DFHZC6920—this error message indicates a problem with your autoinstall control program.
- DFHZC6921—this error message indicates that an unknown APPC device has attempted to connect to CICS, and the the autoinstall control program has given a nonzero return code indicating that the install cannot go ahead. An exception trace entry is made; see the *CICS/ESA Diagnosis Reference* for details of the new trace points.
- DFHZC6922—this error message indicates a problem with the connection model definition CICS is using as a template.
- DFHZC6923—this error message indicates that CICS has received a BIND from an unknown APPC node. The autoinstall process was initiated, but an invalid bind parameter has been detected.

For details of all the new, changed, and obsolete messages, see the *CICS/ESA Migration Guide*.

Changed messages

The following messages are changed:

- DFHZC6935—this message is changed to refer to CONNECTION or TERMINAL. It previously referred to only TERMINAL.
- DFHZC6966—this message is changed to refer to CONNECTION or TERMINAL. It previously referred to only TERMINAL.
- DFHZC2411—14 new instances are added to this message. Each new instance is connected with an exception trace.

Chapter 16. CICS/ESA support for MVS workload management

This chapter describes how CICS uses the MVS workload management feature of MVS/ESA Version 5 Release 1. It covers the following topics:

- Overview
- Benefits of MVS workload management
- Requirements for MVS workload management
- Changes to CICS externals
- Problem determination.

For more information about MVS workload manager, see the *MVS/ESA Planning: Workload Management* manual, GC28-1493.

Overview

MVS/ESA 5.1 provides an improved means of managing sysplex resources across MVS subsystems. This facility, the MVS workload manager, provides automatic, dynamic, balancing of system resources (central processors and storage) across a sysplex by:

- Adopting a goal-oriented approach
- Gathering real-time data from the subsystems that reflect performance at an individual task level
- Monitoring MVS- and subsystem-level delays and waits that are contributing to overall task execution times
- Dynamically managing the sysplex's resources, using the performance goals, and the real-time performance and delay data, as inputs to system resource management algorithms.

This is particularly significant in a sysplex environment, but is also of value to subsystems running in a single MVS image.

To help you migrate to goal-oriented workload management, you can run any MVS image in a sysplex in **compatibility mode**, using the performance management tuning methods of releases of MVS before MVS/ESA 5.1.

Note: If you use CICSPlex SM to control dynamic transaction routing in a sysplex, you can base its actions on the CICS response time goals of the CICS transactions as defined to the MVS workload manager.

MVS workload management terms

The following terms are used in the description of MVS workload management:

classification rules
The rules workload management and subsystems use to assign a service class and, optionally, a reporting class to a work request (transaction). A classification rule consists of one or more of the following work qualifiers:

subsystem type
subsystem instance
userid

accounting information
transaction name
transaction class
source LU, NETID, and LU name

compatibility mode
A workload management mode for an MVS image in a sysplex using the pre-workload management MVS performance tuning definitions from the IEAICSxx and IEAIPSxx members of the SYS1.PARMLIB library.

goal mode
A workload management mode for an MVS image in a sysplex using an MVS workload management service definition to automatically and dynamically balance its system resources according to the active service policy for the sysplex.

report class
Work for which reporting information is collected separately. For example, you can have a report class for information combining two different service classes, or a report class for information on a single transaction.

service class
A subset of a workload having the same service goals or performance objectives, resource requirements, or availability requirements. For workload management, you assign a service goal to a service class.

service definition
An explicit definition of all the workloads and processing capacity in a sysplex. A service definition includes service policies, workloads, service classes, resource groups, and classification rules.

service policy
A set of performance goals for all MVS images using MVS workload manager in a sysplex. There can be only one active service policy for a sysplex, and all subsystems in goal mode within that sysplex process towards that policy. However, you can create several service policies, and switch between them to cater for the different needs of different processing periods.

workload
Work to be tracked, managed and reported as a unit. Also, a group of service classes.

workload management mode
The mode in which workload management manages system resources in an MVS image within a sysplex. The mode can be either compatibility mode or goal mode.

Span of workload manager operation

MVS workload manager operates across a sysplex. You can run each MVS image in the sysplex in either goal mode or compatibility mode. However, there can be only one active service policy for all MVS images running in goal mode in a sysplex.

All CICS regions (and other MVS subsystems) running on an MVS image with MVS workload manager are subject to the effects of workload management.

If the CICS workload involves non-CICS resource managers, such as DB2 and DBCTL, CICS can pass information through the resource manager interface (RMI[7]) to enable MVS workload manager to relate the part of the workload within the non-CICS resource managers to the part of the workload within CICS.

CICS does not pass information across ISC links to relate the parts of the task execution thread on either side of the ISC link. If you use tasks that communicate across ISC links, you must define separate performance goals, and service classes, for the parts of the task execution thread on each side of the ISC link. These rules apply to ISC links that are:

- Within the same MVS image (so called “intrahost ISC”)
- Between MVS images in the same sysplex (perhaps for compatibility reasons)
- Between MVS images in different sysplexes

If you use tasks that communicate across ISC links between two sysplexes, the separate performance goals are defined in the active service policy for each sysplex.

Defining performance goals

You can define performance goals, such as internal response times, for CICS (and other MVS subsystems that comprise your workload). As an alternative to defining your own goals, you can use “discretionary goals”—the workload manager decides how best to run work for which this type of goal is specified. You can define goals for:

- Individual CICS regions
- Groups of transactions running under CICS
- Individual transactions running under CICS
- Transactions associated with individual userids
- Transactions associated with individual LU names
- Transactions associated with individual network identifiers.

The service level administrator defines your installation’s performance goals based on business needs and current performance. The complete definition of workloads and performance goals is called a **service definition**. You may already have this kind of information in a service level agreement (SLA).

You should record the details of your planned service definition on worksheets, as described in the *MVS/ESA Planning: Workload Management* manual. MVS/ESA 5.1 provides an ISPF panel-based application for setting up and adjusting the service definition.

Workload management also collects performance and delay data for work defined by the service definition; this data can be used by reporting and monitoring products, such as the Resource Measurement Facility (RMF), the Service Level Reporter (SLR), Enterprise Performance Data Manager/MVS (EPDM), or vendor products.

---

[7] The CICS interface modules that handle the communication between a task-related user exit and the resource manager are usually referred to as the resource manager interface (RMI) or the task-related user exit (TRUE) interface.

Determining CICS response times before defining goals

Before you set goals for CICS work, you can determine CICS current response times by running CICS in compatibility mode with an arbitrary goal. For this purpose, use the SRVCLASS parameter, provided by MVS/ESA 5.1 in the installation control specification (ICS). This parameter lets you associate a service class with a report performance group, to be run in compatibility mode. You would then:

1. Define a service policy, with a default service class, or classes, for your CICS work, and specify an arbitrary response time goal (say 3 seconds).
2. Define classification rules for the service class or classes.
3. Install the service definition.
4. Activate the service policy in compatibility mode.

The average response time for work within the service classes are reported under the report performance group in the RMF Monitor I workload activity report.

This information helps you to set realistic goals for running your CICS work when you switch to goal mode. The reporting data produced by RMF reports:

- Is organized by service class
- Contains reasons for any delays that affect the response time for the service class (for example, because of the actions of a resource manager or an I/O subsystem).

From the reported information, you may be able to determine configuration changes to improve performance.

Example of using SRVCLASS parameter of IEAICSxx: To obtain CICS response time information while in compatibility mode, you can set up the following:

- In your service definition, set up the following:
  - A test policy, comprising the following:

```
Service Policy Name . . . : CICSTEST
Description . . . . . : Migration (compatibility) mode
```

  - A workload definition, in which to define the required service class:

```
Workload Name  . . . . . : CICSALL
Description  . . . . . . . . CICSTEST migration workload
```

  - A service class for all CICS transactions:

```
Service Class Name . . . . . : CICSALL
Description  . . . . . . . . . All CICS transactions
Workload Name  . . . . . . . . CICSALL
        ---Period---  ---------------------Goal---------------------
Action  #  Duration   Imp.  Description
  __    1              1    Average response time of 00:00:03.000
```

    Note: It does not matter what goal you specify, since it is not used in compatibility mode, but it cannot be discretionary.

  - Specify the name of the service class under the classification rules for the CICS subsystem:

```
Subsystem Type . . . . . : CICS
Default Service Class  . . : CICSALL
```

- In your ICS member in SYS1.PARMLIB (IEAICS*xx*), specify:

  ```
  SUBSYS=CICS,
  SRVCLASS=CICSALL,RPGN=100
  ```

- Install the workload definition in the coupling facility.
- Activate the test service policy, either by using options provided by the WLM ISPF application, or by issuing the following MVS command:

  ```
  VARY WLM,POLICY=CICSTEST
  ```

You receive response time information about CICS transactions in the RMF Monitor I Workload Activity Report under report performance group 100. For more information about defining performance goals and the use of SRVCLASS, see the *MVS/ESA Planning: Workload Management* manual.

Service definitions

You define one service definition for each sysplex. A service definition consists of:

Service policies
You can have one or more service policies, which are a named set of performance goals meant to cover a certain operating period.

If you have varying performance goals, you can define several service policies.

You can activate only one service policy at a time for the whole sysplex, and, when appropriate, switch to another policy.

Workloads
A workload comprise units of work that share some common characteristics that makes it meaningful for an installation to manage or monitor as a group. For example, all CICS work, or all CICS order entry work, or all CICS development work.

A workload is made up of one or more service classes.

Service classes
These are categories of work, within a workload, to which you can assign performance goals.

You can create service classes for groups of work with similar:

- Performance goals

  You can assign the following performance goals to the service classes:

  Response time
  You can define an average response time (the amount of time required to complete the work) or a response time with percentile (a percentage of work to be completed in the specified amount of time).

  Discretionary
  You can specify that the goal is discretionary for any work for which you do not have specific goals.

  Velocity
  For work not related to transactions, such as batch jobs and started tasks. For CICS regions started as started tasks, a velocity goal applies only during start-up.

Notes:

1. For service classes for CICS transactions, you cannot define velocity performance goals, discretionary goals, or multiple performance periods.
2. For service classes for CICS regions, you cannot define multiple performance periods.

- Business importance to the installation

  You can assign an importance to a service class, so that one service class goal is recognized as more important than other service class goals. There are five levels of importance, numbered, from highest to lowest, 1 to 5.

You can also create service classes for started tasks and JES, and can assign resource groups to those service classes. You can use such service classes to manage the workload associated with CICS as it starts up, but before CICS transaction-related work begins. (Note that when you define CICS in this way, the address space name is specified as TN, for the task or JES "transaction" name.)

There is a default service class, called SYSOTHER. It is used for CICS transactions for which MVS workload management cannot find a matching service class in the classification rules—for example, if the couple data set becomes unavailable.

Classification rules

These rules determine how to associate incoming work with a service class. Optionally, the classification rules can assign incoming work to a report class, for grouping report data.

There is one set of classification rules for each service definition. The classification rules apply to every service policy in the service definition; so there is one set of rules for the sysplex.

You should use classification rules for every service class defined in your service definition.

Classification rules categorize work into service classes and, optionally, report classes, based on work qualifiers. You set up classification rules for each MVS subsystem type that uses workload management. The work qualifiers that CICS can use (and which identify CICS work requests to workload manager) are:

LU	LU name
LUG	LU name group
NET	Network identifier
NETG	Network identifier group
SI	Subsystem instance (VTAM applid)
SIG	Subsystem instance group
TN	Transaction identifier
TNG	Transaction identifier group
UI	Userid
UIG	Userid group

Notes:

1. You should consider defining workloads for terminal-owning regions only. Work requests do not normally originate in an application-owning region. They (transactions) are normally routed to an application-owning region from a terminal-owning region, and the work request is classified in the terminal-owning region. In this case, the work is not reclassified in the application-owning region.

   If work originates in the application-owning region it is classified in the application-owning region; normally there would be no terminal.

2. You can use identifier group qualifiers to specify the name of a group of qualifiers; for example, GRPACICS could specify a group of CICS tranids, which you could specify on classification rules by TNG GRPACICS. This is a useful alternative to specifying classification rules for each transaction separately.

You can use classification groups to group disparate work under the same work qualifier—if, for example, you want to assign it to the same service class.

You can set up a hierarchy of classification rules. When CICS receives a transaction, workload manager searches the classification rules for a matching qualifier and its service class or report class. Because a piece of work can have more than one work qualifier associated with it, it may match more than one classification rule. Therefore, the order in which you specify the classification rules determines which service classes are assigned.

Note: You are recommended to keep classification rules simple.

Example of using classification rules: As an example, you might want all CICS work to go into service class CICSB except for the following:

- All work from LU name (terminal identifier) S218, except the PAYR transaction, is to run in service class CICSA
- Work for the PAYR transaction (payroll application) entered at LU name S218 is to run in service class CICSC.
- All work from terminals other than S218, and whose termids begin with S2, is to run in service class CICSD.

You could specify this by the following classification rules:

```
Subsystem Type . . . . . . . CICS

    -------Qualifier-----------             --------Class---------
    Type       Name     Start               Service     Report
                                 DEFAULTS: CICSB       ________
1   LU         S218                       CICSA       ________
2     TN         PAYR                     CICSC       ________
1   LU         S2*                        CICSD       ________
```

Note: In this classification, the PAYR transaction is nested as a sub-rule under the classification rule for terminal S218, indicated by the number 2, and the indentation of the type and name columns.

Consider the effect of these rules on the following work requests:

```
                Request 1    Request 2    Request 3    Request 4

LU name ......  S218         A001         S218         S214
Transaction ..  PAYR         PAYR         DEBT         ANOT
```

- For request 1, the work request for the payroll application runs in service class CICSC. This is because the request is associated with the terminal with LU name S218, and the TN—PAYR classification rule specifying service class CICSC is nested under the LU—S218 classification rule qualifier.
- For request 2, the work request for the payroll application runs in service class CICSB, because it is ***not*** associated with LU name S218, nor S2*, and there are no other classification rules for the PAYR transaction. Likewise, any work requests associated with LU names that do not start with S2 run in service class CICSB, as there are classification rules for LU names S218 and S2* only.
- For request 3, the work request for the DEBT transaction runs in service class CICSA, because it is associated with LU name S218, and there is no DEBT classification rule nested under the LU—S218 classification rule qualifiers.
- For request 4, the work request for the ANOT transaction runs in service class CICSD, because it is associated with an LU name starting S2, but not S218.

However, if the classification rules were specified as:

```
1 TN         PAYR                      CICSA      ________
1 LU         S218                      CICSA      ________
2   TN       PAYR                      CICSC      ________
1 LU         S2*                       CICSD      ________
```

the PAYR transaction would always run in service class CICSA, even if it were associated with LU name S218.

Using a service definition base: To minimize the amount of data you need to enter into the ISPF workload application, you use a **service definition base**. When you set up your service definition, you identify the workloads, the service classes, and their goals, based on your performance objectives. Then you define classification rules. This information makes up the service definition base. The base contains workloads, service classes, resource groups, report classes, and classification rules.

All workloads, service classes, and classification rules defined in a service definition base apply to every policy that you define. If you do not have any other business requirements to modify a service goal or a resource group from the service definition base, you can run an installation with one policy.

MVS performance definitions

Running CICS on MVS/ESA 5.1 or later, enables you to use the MVS workload management facility that MVS/ESA 5.1 provides for managing sysplex resources across MVS subsystems, in parallel with the existing system resource management facilities.

For information about MVS workload management, see the MVS publication *Planning: Workload Management*, GC28-1493.

If you want to use the MVS workload manager facility, you should:

1. Implement workload management on the MVS images that the CICS workload is to run on, as outlined in "Implementing MVS workload management."
2. Ensure that CICS performance parameters correspond to the policies defined for MVS workload management, as outlined in "Matching CICS performance parameters to service policies" on page 284.

If you have not installed MVS/ESA 5.1, or do not want to use the MVS workload management facility, you should review your MVS performance definitions to ensure that they are still appropriate for CICS/ESA 4.1. To do this, review parameters in the IEAICS and IEAIPS members of the MVS PARMLIB library. For more information about these MVS performance definitions, see the *&esasplitc.*.

Implementing MVS workload management

The task of implementing MVS workload management is part of the overall task of planning for, and installing, MVS/ESA 5.1.

Implementing MVS workload management generally involves the following steps:

1. Establish your workloads.
2. Set your business priorities.
3. Understand your performance objectives.
4. Define critical work.
5. Define performance objectives based on current:
   - Business needs
   - Performance:
     - Reporting and monitoring products
     - Capacity planning tools
     - IEAICS and IEAIPS parameters.
6. Get agreement for your workload performance objectives.
7. Specify a service level agreement or performance objectives.
8. Specify an MVS WLM service definition using the information from step 7.

   Note: It is helpful at this stage to record your service definition in a form that will help you to enter it into the MVS workload manager ISPF application. You are recommended to use the worksheets provided in the MVS publication *Planning: Workload Management*.
9. Install MVS/ESA 5.1.
10. Set up a sysplex with a single MVS image, and run in workload manager compatibility mode.

11. Upgrade your existing XCF couple data set.
12. Start the MVS workload manager ISPF application, and use it in the following steps.
13. Allocate and format a new couple data set for workload management. (You can do this from the ISPF application.)
14. Define your service definition.
15. Install your service definition on the couple data set for workload management.
16. Activate a service policy.
17. Switch the MVS image into goal mode.
18. Start up a new MVS image in the sysplex. (That is, attach the new MVS image to the couple data set for workload management, and link it to the service policy.)
19. Switch the new MVS image into goal mode.
20. Repeat steps 18 and 19 for each new MVS image in the sysplex.

Notes:

1. CICS/ESA 4.1 support for MVS workload manager is initialized automatically during CICS startup.
2. All CICS regions (and other MVS subsystems) running on an MVS image with MVS workload management are subject to the effects of workload manager.

Matching CICS performance parameters to service policies

You must ensure that the CICS performance parameters are compatible with the workload manager service policies used for the CICS workload.

In general, you should define CICS performance objectives to the MVS workload manager first, and observe the effect on CICS performance. Once the MVS workload manager definitions are working correctly, you can then consider tuning the CICS parameters to further enhance CICS performance. However, you should use CICS performance parameters as little as possible.

Performance attributes that you might consider using are:

- Transaction priority, passed on dynamic transaction routing. (Use prioritization carefully, if at all.) The priority assigned by the CICS dispatcher must be compatible with the task priority defined to MVS workload manager.
- Maximum number of concurrent user tasks for the CICS region.
- Maximum number of concurrent tasks in each transaction class.

Activating CICS support for MVS workload manager

CICS/ESA 4.1 support for MVS workload manager is initialized automatically during CICS startup.

Customer-written resource managers and other non-CICS code which is attached to CICS via the RMI must be modified to provide workload manager support, if workload manager is to work optimally for CICS-based tasks which cross the RMI into such areas.

Benefits of MVS workload management

The benefits of using MVS workload manager are:

- Improved performance through MVS resource management

  The improvement is likely to depend on many factors, for example:

  – System hardware configuration

  – The way the system is partitioned

  – Whether CICS subsystems are single or multi-region

  – The spread of types of applications or tasks performed, and the diversity of their profile of operation

  – The extent to which the sysplex workload changes dynamically

- Improved efficiency of typical MVS sysplexes

  – Improved overall capacity
  – Increased work throughput.

- Simplified MVS tuning

  Generally, for those systems where it is difficult to attain or maintain optimal tuning, or where it is time consuming to achieve by current means, will tend to obtain the greater benefit.

The main benefit is that you no longer have to continually monitor and tune CICS to achieve optimum performance. You set your workload objectives in the service definition, and let workload management component of MVS manage the resources and the workload to achieve your objectives.

The MVS workload manager produces performance reports that you can use to establish reasonable performance goals and for capacity planning.

Requirements for MVS workload management

To use MVS workload management you need the following software:

- MVS/ESA System Product (MVS/ESA SP) - JES2 Version 5 Release 1 or a later, upward-compatible, release
- MVS/ESA System Product (MVS/ESA SP) - JES3 Version 5 Release 1 or a later, upward-compatible, release

For MVS workload manager operation across the CICS task-related user exit interface to other subsystems, such as DB2 and DBCTL, you need the appropriate releases of these products.

Resource usage

CICS support for MVS workload management has a negligible effect on either CICS use of storage or on CICS performance.

Changes to CICS externals

CICS support for MVS workload management introduces a number of changes to CICS externals. These are:

- Changes to task-related user exits
- Changes to the exit programming interface (XPI)

Changes to task-related user exits

General-use programming interface

A new parameter, UEPPBTOK, is added to the end of standard DFHUEPAR parameter list for task-related user exits. It is placed at the end of the parameter list, following UEPTIND to ensure compatibility with the parameter lists of earlier releases.

UEPPBTOK in DFHUEPAR

The UEPPBTOK parameter is described as follows:

UEPPBTOK Address of the performance block token used for workload management, to enable resource managers to relate their own performance blocks for the work request with the original CICS performance block. For example, DBCTL and DB2 need to correlate the work they do on behalf of CICS with the originating CICS task, so that MVS workload manager can measure the performance of the whole CICS task.

End of General-use programming interface

Changes to the exit programming interface

Product-sensitive programming interface

The CICS exit programming interface is enhanced for MVS workload management by the addition of a new parameter, WLM_WAIT_TYPE, to the CICS dispatcher functions SUSPEND and WAIT_MVS.

The SUSPEND call

The CICS dispatcher SUSPEND function suspends execution of a running task. The task can be resumed in one of two ways. You can issue the XPI RESUME function call, or CICS resumes the task automatically if the INTERVAL value that you specify on the SUSPEND function of DFHDSSRX macro expires. Suspended tasks can also be purged by the operator, or by an application program, or by the deadlock time-out facility.

SUSPEND

```
DFHDSSRX [CALL,]
         [CLEAR,]
         [IN,
          FUNCTION(SUSPEND),]
          ...
          [WLM_WAIT_TYPE(name1),]
          [OUT,
          ...
          REASON(name1 | *)]
```

WLM_WAIT_TYPE(name1)
specifies, in a 1-byte location, the reason for suspending the task. This indicates the nature of the task's wait state to the MVS workload manager.

The equated values for the type of wait are as follows:

CONV
Waiting on a conversation.

CMDRESP
Waiting on a command response.

DISTRIB
Waiting on a distributed request.

IDLE
A CICS task, acting as a work manager, that has no work request that is allowed to service within the monitoring environment. For example, journaling code that suspends itself when there are no journaling I/O operations to perform.

IO Waiting on an I/O operation or indeterminate I/O-related operation (locks, buffer, string, and so on).

LOCK
Waiting on a lock.

MISC
Waiting on an unidentified resource.

Note: This is the default reason given to the wait if you suspend a task and do not specify the WLM_WAIT_TYPE parameter.

OTHER_PRODUCT
Waiting on another product to complete its function; for example, when the workload has been passed to DB2.

SESS_LOCALMVS
Waiting on the establishment of a session in the MVS image on which this CICS region is running.

SESS_NETWORK
Waiting on the establishment of a session elsewhere in the network (that is, not on this MVS image).

SESS_SYSPLEX
Waiting on establishment of a session somewhere in the sysplex (that is, not on this MVS image).

TIMER
Waiting on the timeout of a timer (for example, a task that suspends itself for a specified time).

If you are running CICS in an MVS goal-mode workload management environment (that is, you are using goal-oriented performance management), you are recommended to specify the reason for suspending the task on the WLM_WAIT_TYPE parameter.

The WAIT_MVS call

WAIT_MVS requests a wait on an MVS event control block (ECB) or on a list of MVS ECBs. For example, you can issue the WAIT_MVS to wait for completion of an MVS task for which you have issued ATTACH and provided a task-completion ECB.

The CICS dispatcher does not clear the ECBs when a WAIT_MVS request is received. If any ECB is already posted, control is returned immediately to the exit program with a response of 'OK'.

A single ECB must not be the subject of more than one wait at a time. If any ECB is already being waited on when a WAIT_MVS request is received, the request is rejected. The RESPONSE code is 'DSSR_INVALID', and the REASON code 'DSSR_ALREADY_WAITING'.

Note: ECBs used in WAIT_MVS requests must never be "hand posted". They must be posted using the MVS POST macro.

WAIT_MVS

```
DFHDSSRX [CALL,]
         [CLEAR,]
         [IN]
         [FUNCTION(WAIT_MVS),
         ...
          [WLM_WAIT_TYPE(name1),]
         ...
         REASON(name1 | *)]
```

WLM_WAIT_TYPE(name1)
specifies, in a 1-byte location, the reason for suspending the task. This indicates the task's wait state to the MVS workload manager. For details of the various types of wait that you can specify, see the description of WLM_WAIT_TYPE on the SUSPEND call of the DFHDSSRX macro on page 287.

End of Product-sensitive programming interface

Problem determination

There are some new CICS messages that are issued during CICS initialization if problems are encountered when CICS tries to connect to MVS workload manager.

For details of all the new, changed, and obsolete messages in CICS/ESA 4.1, see the *CICS/ESA Migration Guide*.

There are changes to diagnosis methods for investigating performance problems relating to work requests running under CICS on MVS images where workload manager is operative. When analyzing such circumstances, you need to consider the workload manager operation, the different methods of system tuning implied by the use of workload manager, and any interaction between the tuning parameters of CICS and workload manager.

When resolving wait problems, note the new WLM_WAIT_TYPE parameter of the DFHDSSRX macro call.

Chapter 17. Install and discard global user exit

This chapter describes the new global user exit, XRSINDI. It covers the following topics:

- Overview
- Benefits of using the new global user exit
- Changes to CICS externals.

This chapter contains Product-sensitive Programming Interface information.

Overview

For efficient management of CICS regions, systems management products, such as CICSPlex System Manager/ESA (CICSPlex SM), need to have up-to-date information about the resources installed in any given region. A new global user exit, XRSINDI, is introduced to enable this information to be obtained by allowing the systems management program to monitor the additions and deletions of resources. For this purpose, the global user exit program is invoked immediately after CICS successfully installs or discards a resource.

See “The XRSINDI global user exit” on page 292 for details of all the activities that cause this new exit to be driven.

Benefits of the install discard global user exit

This global user exit provides a systems management product, such as CICSPlex SM, with an exit immediately after the successful install or discard of a resource.

The XRSINDI global user exit is directly concerned with the management of CICS resources, and allows a systems management product to maintain up-to-date information about which resources are installed in any given region.

Using XRSINDI to achieve improved systems management by efficiently maintaining an accurate resource database also provides an aid to problem diagnosis.

Trace points help you determine whether an error has occurred within the program enabled at the XRSINDI global user exit point.

This exit point is added to the resource managers of the following resources:

- Autoinstall terminal models
- Connections
- Files
- All FEPI resources
- Mapsets
- Modegroups
- Partitionsets
- Partners
- Programs
- Profiles

- Sessions
- Terminals
- Transactions
- Transaction classes.

Changes to CICS externals

The only change to CICS externals to support the monitoring of the install and discard functions is the addition of a new global user exit, XRSINDI.

The exit programming interface is not changed: XRSINDI obeys the same rules as all other global user exits.

The XRSINDI global user exit

The XRSINDI global user exit is driven, if it is enabled, immediately after CICS successfully installs or discards a resource definition.

The install and discard activities that drive the exit are as follows:

- The install function of the group list on a cold start of CICS
- The CEDA INSTALL command
- All autoinstall operations, as follows:
  - The autoinstall of a terminal, connection, program, mapset, or partitionset
  - The automatic discard of an unused terminal, controlled by the AILDELAY system initialization parameter and the SIGNOFF parameter on the TYPETERM resource definition.
- A CEMT DISCARD and EXEC CICS DISCARD command
- The front-end programming interface (FEPI) install and discard operations: the EXEC CICS FEPI INSTALL command and EXEC CICS FEPI DISCARD command.

The parameter list is designed to pass the names of more than one resource installed or discarded, in field UEPIDNAM. When designing your global user exit program, do not assume that the number of resource names passed is always one. You are recommended to analyze the resources within a loop based on the value referenced by UEPIDNUM.

Note that the names of modegroups are prefixed with the corresponding connection name. There is no separator between the two names: the first four characters form the connection name, followed by eight characters for the modegroup. The parts of the concatenated name are fixed length—if connection names are defined with less than four characters, they are padded with blanks in the concatenated names. Similarly, the connection names for a front-end programming interface (FEPI) connection is a concatenation of a FEPI node name and a FEPI target name, each of which is 8 characters long (fixed length) with no separator.

The exit is driven once for each individual resource in a group list installed during a CICS cold start. If you are concerned about the performance overhead on a cold start, you should not enable the exit until after the group list is installed. To obtain the information about resources installed prior to enabling the exit, you can write a

program to scan the tables of installed resources, using the EXEC CICS INQUIRE *resource_name* browse function.

Exit XRSINDI

	Invoked whenever CICS installs or discards a resource definition.	
Exit-specific parameters	**UEPTRANID**	Address of the 4-byte transaction ID
	UEPUSER	Address of the 8-byte user ID
	UEPTERM	Address of the 4-byte terminal ID
	UEPPROG	Address of the 8-byte application program name
	UEPIDREQ	Address of the 1-byte install or discard identifier. The values are:
		UEIDINS This request is for an install.
		UEIDDIS This request is for a discard.
	UEPIDTYP	Address of the 1-byte type of resource. The values are:
		UEIDTRAN A transaction
		UEIDPROF A profile
		UEIDPROG A program
		UEIDMAP A mapset
		UEIDPSET A partitionset
		UEIDTERM A terminal
		UEIDCONN A connection
		UEIDMODE A modegroup
		UEIDSESS A session
		UEIDFILE A file
		UEIDPART A partner
		UEIDTCLS A transaction class
		UEIDAITM An autoinstall terminal model
		UEIDFECO A FEPI connection
		UEIDFENO A FEPI node
		UEIDFEPO A FEPI pool
		UEIDFEPS A FEPI propertyset
		UEIDFETA A FEPI target

Exit XRSINDI (continued)

Exit-specific parameters (continued)	**UEPIDLEN**	Address of the length of an individual resource name, as a full-word binary value.
	UEPIDNUM	Address of the number of resources reported by this call, as a full-word binary value.
	UEPIDNAM	Address of a variable-length list containing the names of the individual resources reported by this call.
	UEPIDREC	Address of a 1-byte identifier indicating whether resources are recovered at a warm or emergency restart. The values are: **UEIDKEEP** The resources are recoverable at a warm or emergency restart. **UEIDLOSE** The resources are not recoverable. **Note:** The exit is not driven during a CICS restart.
Return codes	**UERCNORM** Continue processing. This is the default.	
	UERCPURG Task purged during XPI call.	
XPI calls	You can use all XPI calls.	
API and SPI commands	You can use all API and SPI commands, but you must take care to avoid recursion. Any command that can potentially cause CICS to install a resource (such as EXEC CICS LOAD, for example), causes a recursive invocation of the exit. You must ensure that you write your global user exit program to handle a recursive condition.	

Important

Abends in a program enabled at the XRSINDI exit point may cause CICS to terminate, because for some resources the exit is driven during syncpoint. If the exit returns code UERCPURG during syncpoint for these resources, abend code AUEP is produced and CICS terminates.

Chapter 18. Monitoring and statistics

This chapter describes the monitoring and statistics changes in CICS/ESA 4.1. It covers the following topics:

- Overview
- Benefits
- Changes to CICS externals.

Overview

There have been a number of additions and improvements to both performance and exception class records in this release. These include:

- Significant additions and improvements to the detailed level of performance data records.
- Monitoring data for the front end programming interface (FEPI) has been incorporated. The data recorded includes:
  - The type and number of requests made to FEPI
  - The time spent waiting for requests to FEPI to complete
  - The number of requests to FEPI that are timed-out.
- The provision of additional performance class records for long running tasks, conversational tasks, and by unit of work.
- Allow user-specification of the 4-byte SYSEVENT record subsystem name.
- Allowing you to request the timestamp fields as either GMT or local time.
- Improved correlation for CICS SMF data with SMF data from other products such as RMF. This is achieved by adding JOB related information, such as jobname to the SMF Product Section of the CICS SMF 110 records. This enables monitoring utilities to have improved reporting facilities.
- Reduced monitoring control table (MCT) definition requirements, with some monitoring options now specified as system initialization parameters. The ability to change dynamically these monitoring options via SPI and CEMT commands.

There are many additions, changes, and modifications to CICS statistics in this release. They are aimed primarily at improving the way that you can collect statistics so that you have easier offline performance analysis and tuning, and better data correlation. There are improvements in the consistency of the last reset time and the use of local time. The utility program (DFHSTUP) is enhanced to provide clearer layout and resource type selectivity. As with monitoring, there are changes in the way the statistics are reported and correlated with other SMF data so that your reporting utilities can reflect the information gathered from your whole system.

The generation and collection of FEPI statistics are now collected at the specified statistics interval.

Monitoring

There are additions, changes, and deletions to CICS/ESA monitoring data, which are summarized in the following sections:

Additions

- Additions and improvements to the performance class data:
  - First dispatch delay
  - TRANCLASS dispatch delay time
  - MXT dispatch delay time
  - Task ENQ delay time
  - Transaction class name
  - LU6.1 I/O wait time
  - LU6.2 I/O wait time
  - LU6.2 message and character counts
  - Transaction routing sysid
  - Resource manager interface (RMI)—elapsed and suspend time suspend time.
- Additional performance class records for long running tasks and by unit-of-work. These provide you with:
  - The ability to request performance class records for long-running transactions
  - The ability to select performance class records by unit-of-work (syncpoint).
- The option of online reporting of the monitoring timestamps in LOCAL time rather than GMT, with GMT still the default.
- New sample monitoring control tables (MCTs) are provided for a terminal owning region (TOR), an application owning region (AOR), and a file owning region (FOR). These show you the types of fields that can be excluded in order to reduce the size of the performance class record output by CICS monitoring. Using these sample MCTs you can reduce the size of a performance record by 208 bytes for a TOR, by 48 bytes for an AOR, and 232 bytes for an FOR.
- Improvements to the exception class record by the addition of the transaction class name.

Changes

- The are changes to the following fields of performance class data:
  - Transaction type
  - Record type
  - Principal facility message and character counts.
- The maximum size of user event monitoring point (EMP) character fields is increased from 256 bytes to 8KB and the maximum user area size is increased from 4KB to 16KB.
- The performance class data field ‘exception wait time’ is updated when exception conditions are encountered even when the exception class is inactive.
- The I/O wait time for LU6.1 sessions has been separated from the current TC I/O wait time field into a new performance class data field.
- Changes to the transaction type field in exception class data.

- The granularity of wait-on-string exceptional condition for files is improved by separating the string waits into two separate exception resource types LSRPOOL and FILE.
- The temporary storage exception record(s) is improved by including the TS queue name in the resource id field of the exception record.

Deletions

- The following performance class data fields are removed:
  - Operator-id
  - Program-storage high-water-mark (UDSA and EDSA)
  - Tclass
  - Terminal storage.
- The following exception class data fields are removed in this release:
  - Operator-id
  - Tclass.

Statistics

There are additions, changes and deletions to statistics records, which are described in the following sections.

Additions and changes

- There are improvements and additions to allow easier offline performance analysis and tuning, and data correlation.
- There are additional and changed statistics in the following areas:
  - Terminal-autoinstall statistics
  - DBCTL statistics
  - Dispatcher statistics
  - DL/I statistics
  - FEPI pool statistics
  - FEPI connection statistics
  - FEPI target statistics
  - File control statistics
  - ISC/IRC system and mode entry statistics
  - Journal control statistics
  - Loader statistics
  - LSR pool statistics
  - Program-autoinstall statistics
  - Storage manager statistics
  - Terminal control statistics
  - Terminal autoinstalled statistics
  - Transaction statistics
  - Transaction class statistics
  - Transaction manager statistics
  - Transient data statistics
  - User domain statistics
  - VTAM statistics.
- Times are synchronized between offline and online reporting of statistics at the last reset time.
- Statistical timestamps are now provided in both GMT and local time.

- Improved readability of the file control report produced by the statistics utility program (DFHSTUP).

Deletions

- The following are replaced by the new transaction manager statistics:
  - Task control statistics
  - Transaction statistics
  - Tclass statistics
- The domain manager statistics are removed.

Benefits

There many benefits provided by the changes and enhancements to monitoring and statistics in CICS/ESA 4.1. In general terms, the CICS/ESA 4.1 monitoring and statistics facilities are designed to:

- Improve your ability to manage more effectively your CICS systems.
- Simplify the control of your monitoring control table data. For example, you no longer have to specify the CPU option in the monitoring control table (MCT), as CPU time is always measured.
- Enable you to change monitoring options dynamically, using the CEMT INQUIRE and SET MONITOR, and using EXEC CICS INQUIRE and SET MONITOR commands.
  - Separate performance class records for conversational tasks.
  - Separate performance class records by unit of work.
  - Set the frequency at which monitoring generates performance class records for long-running units of work.
- Improve readability of the file control statistics formatted by the statistics utility program (DFHSTUP).
- Provide monitoring and statistics support for FEPI:
  - The monitoring and statistics data can help with performance tuning and resource planning for applications using FEPI.
  - The usage and the performance of FEPI can be compared with that of other CICS components.
  - Data about FEPI usage now appears in the standard statistics reports.
- Provide a selective report of DFHSTUP by resource types.

Changes to CICS externals

The general changes to CICS monitoring and statistics, and incorporation of the FEPI feature into the base product, results in a number of changes to CICS externals: These are:

- Changes to system definition
- Changes to resource definition
- Changes to the system programming interface
- Changes to CICS-supplied transactions
- Changes to monitoring and statistics data.

Changes to system definition

The following monitoring options are added to system initialization parameters.

Table 22 lists the new system initialization parameters which are required for support of monitoring.

Table 22. New DFHSIT macro parameters

	DFHSIT	TYPE={**CSECT**|DSECT} ... [,MNCONV={**NO**|YES}] [,MNFREQ={**0**|hhmmss}] [,MNSUBSYS={**null**|xxxx}] [,MNSYNC={**NO**|YES}] [,MNTIME={**GMT**|LOCAL}] ...
	END	DFHSITBA

MNCONV={NO|YES}
Specifies whether or not conversational tasks are to have separate performance class records produced for each pair of terminal control I/O requests.

Any clock (including user-defined) that is active at the time such a performance class record is produced is stopped immediately before the record is written. After the record is written, such a clock is reset to zero and restarted. Thus a clock whose activity spans more than one recording interval within the conversational task appears in multiple records, each showing part of the time, and the parts adding up to the total time the clock is active. The high-water-mark fields (which record maximum levels of storage used) are reset to their current values. All other fields are set to X'00', except for the key fields (transid, termid). The monitoring converse status is recorded in the CICS global catalog for use during warm and emergency restarts.

MNFREQ={0|hhmmss}
Specifies the interval for which CICS automatically produces a transaction performance class record for any long-running transaction. The monitoring frequency value is recorded in the CICS global catalog for use during warm and emergency restarts.

0 means that no frequency monitoring is active.

hhmmss
is the interval for which monitoring produces automatically a transaction performance class record for any long-running transaction. Specify a 1 to 6 digit number in the range 001500–240000. Numbers that are fewer than six digits are padded with leading zeroes.

MNSUBSYS={null|xxxx}
Specifies the 4-character name to be used as the subsystem identification in the monitoring SYSEVENT class records. If you do not specify a name, the subsystem identification defaults to the first four characters of the CICS generic APPLID. The monitoring subsystem id is recorded in the CICS global catalog for use during warm and emergency restarts.

For more information on the SYSEVENT class of monitoring data and the subsystem identification, and about the implications for SYSEVENT recording in a MVS Workload Manager environment, see the *CICS/ESA Performance Guide*.

MNSYNC={NO|YES}
Specifies whether or not you want CICS to produce a transaction performance class record when a transaction takes an implicit or explicit syncpoint (unit-of-work). No action is taken for syncpoint rollbacks. The monitoring syncpoint status is recorded in the CICS global catalog for use during warm and emergency restarts.

MNTIME={GMT|LOCAL}
Specifies whether you want the time stamp fields in the performance class monitoring data to be returned to an application using the EXEC CICS COLLECT STATISTICS MONITOR(taskno) command in either GMT or local time. The monitoring time value is recorded in the CICS global catalog for use during warm and emergency restarts.

For more information on the EXEC CICS COLLECT STATISTICS command, see the *CICS/ESA System Programming Reference*.

Changes to resource definition

The changes to the monitoring control table for CICS/ESA 4.1 are as follows:

DFHMCT TYPE=INITIAL

The CPU and CONV parameters of the DFHMCT TYPE=INITIAL are removed. The CONV parameter is now specified as a system initialization parameter. The CPU option is no longer required because CPU time is always measured. The TYPE=INITIAL macro for the MCT is now as follows:

```
DFHMCT TYPE=INITIAL
       [,SUFFIX=xx]
```

DFHMCT TYPE=EMP

The MOVE option of the PERFORM parameter of the DFHMCT TYPE=EMP macro definition is enhanced to increase the maximum size of user character fields from 256 bytes to 8192 bytes. The descriptions of these parameters are as follows:

PERFORM=(option|,...|)
This operand must be coded when CLASS=PERFORM is coded. It specifies that information is to be added to or changed in the user fields of a performance class data record by this user event monitoring point (EMP). The user fields for each user are distinguished by a separate name in the ID operand and can comprise:

- Up to 256 counters
- Up to 256 clocks, each made up of a 4-byte accumulator and 4-byte count
- A byte string of up to 8192 bytes.

Note: The maximum size of user data in each performance record is increased from 4096 bytes to 16384 bytes.

Actions will be performed on the user fields according to the options specified. PERFORM can be abbreviated to PER.

MOVE(number3,number4)

A string of data is to be moved into the user byte-string field. To use this option, both the DATA1 and DATA2 fields must be passed from the user EMP.

The user byte-string field is updated starting at the offset specified by number3. The data to be moved starts at the address supplied in the DATA1 field. The maximum length of data that can be moved is given by number4 (in bytes), and the actual length of data that is to be moved is given by the value of the DATA2 field. If the value of DATA2 is zero, then the length of the data given by number4 is moved.

Number3 is a decimal integer in the range 0 to 8191, and number4 is a decimal integer in the range 1 to 8192. The maximum length of the user character field is (number3 + number4), and must be in the range 1 to 8192.

Note: Only one of the MLTCNT and MOVE options can be used in each DFHMCT TYPE=EMP macro instruction.

DFHMCT TYPE=RECORD

DFHFEPI is introduced as a group field name to enable you to control the inclusion or exclusion of FEPI monitoring data as a group. Within the DFHFEPI group are FEPI field numbers to identify the individual FEPI fields.

DFHMCT	TYPE=RECORD ,CLASS=PERFORM ,EXCLUDE=(DFHFEPI)

Changes to the system programming interface

The CICS system programming interface (SPI) is enhanced for monitoring and statistics, with changes to the COLLECT, PERFORM, INQUIRE, and SET commands:

- Additional options on the EXEC CICS COLLECT STATISTICS and the EXEC CICS PERFORM STATISTICS RECORD commands.
- Additional options on the EXEC CICS INQUIRE and SET MONITOR commands.

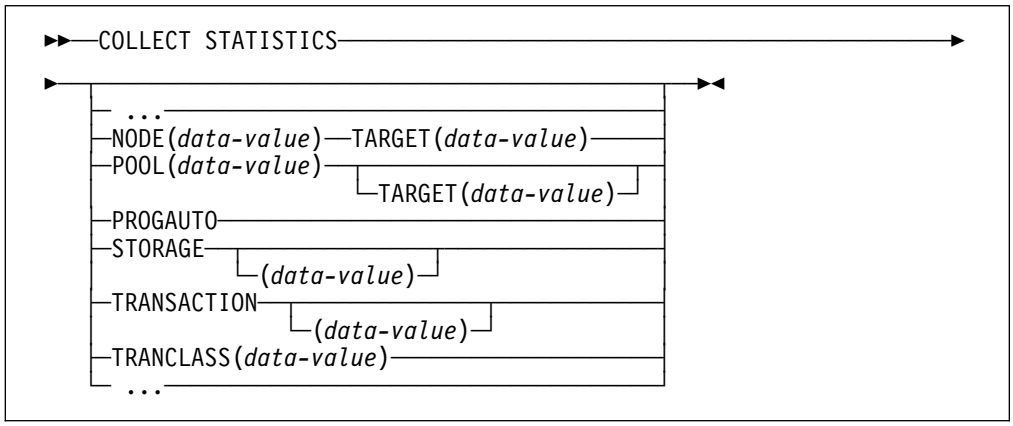
EXEC CICS COLLECT STATISTICS

General-use programming interface

Function

The EXEC CICS COLLECT STATISTICS command returns the current statistics for a named resource or resource type.

Syntax



Resource types: The following list give the resource types available and the restrictions that apply:

- You can request only global statistics for AUTOINSTALL, DISPATCHER, DTB, IRCBATCH, PROGAUTO, TABLEMGR, TSQUEUE, and VTAM.
- You can request either the global statistics or the statistics of a specific resource for MONITOR, PROGRAM, STORAGE, SYSDUMPCODE, TDQUEUE, TRANDUMPCODE and TRANSACTION.
- You can request only the specific statistics for CONNECTION, FILE, JOURNALNUM, LSRPOOL, FEPI POOL, FEPI CONNECTION (TARGET and NODE), FEPI TARGET (POOL and TARGET), TCLASS, TERMINAL, and TRANCLASS.

COLLECT STATISTICS options

PROGAUTO
The program manager autoinstall statistics.

STORAGE
Without the subpool name gives you the storage manager DSA statistics and with the subpool name gives you the storage manager domain subpool statistics for the named subpool. The subpool name is an 8-character data-value.

TRANSACTION(data-value)
specifies the 4-character transaction name. Without the name, CICS gives you the transaction manager (global) statistics and with the name gives you the transaction manager statistics for the named transaction.

TRANCLASS(data-value)
specifies, as an 8-character value, the transaction class identifier. This gives you statistics for the named transaction class.

POOL(data-value) TARGET(data-value)
Resource statistics for a FEPI target within a FEPI pool. POOL is an 8-character data-value specifying a FEPI pool identifier, and TARGET is an 8-character data value identifying the specific FEPI target identifier defined within that POOL.

POOL(data-value)
specifies resource statistics for a FEPI pool.

NODE(data-value) TARGET(data-value)
specifies resource statistics for a FEPI connection. NODE is an 8-character data-value identifying the specific NODE, and TARGET is an 8-character data-value specifying the FEPI target corresponding to the NODE.

Copybooks are provided in ASSEMBLER, COBOL, and PL/I, that map the returned statistics. You can find the copybooks in the following libraries:

ASSEMBLER	CICS410.SDFHMAC
COBOL	CICS410.SDFHCOB
PL/1	CICS410.SDFHPL1

A list of the statistics data copybooks that you can use follows. The names of the copybooks are the same in each language.

Collect statistics command	Copybook	Type of record
AUTOINSTALL	DFHA04DS	Autoinstall statistics (global)
DTB	DFHA05DS	DTB statistics (global)
TERMINAL(name)	DFHA06DS	Terminal control (specific)
LSRPOOL(number)	DFHA08DS	LSRPOOL pool statistics (specific)
TDQUEUE(name)	DFHA10DS	TDQUEUE (specific)
TDQUEUE	DFHA11DS	TDQUEUE (global)
TSQUEUE	DFHA12DS	TSQUEUE statistics (global)
JOURNALNUM(name)	DFHA13DS	Journal control (specific)
CONNECTION(name)	DFHA14DS	ISC/IRC system entry (specific)
TABLEMGR	DFHA16DS	Table manager statistics (global)
FILE(name)	DFHA17DS	File Control (specific)
IRCBATCH	DFHA19DS	IRC Batch statistics (global)
POOL(name)	DFHA22DS	FEPI Pool statistics (specific)
CONNECTION(name) (FEPI)	DFHA23DS	FEPI Connection statistics (specific)
TARGET(name)	DFHA24DS	FEPI Target statistics (specific)
DISPATCHER	DFHDSGDS	Dispatcher statistics (global)
PROGRAM	DFHLDGDS	Loader statistics for programs (global)
PROGRAM(name)	DFHLDRDS	Loader statistics for programs (specific)
MONITOR	DFHMNGDS	Monitoring statistics (global)
MONITOR(name)	DFHMNTDS	Monitoring statistics (specific)
PROGAUTO	DFHPGGDS	Program manager autoinstall statistics (global)
SYSDUMPCODE	DFHSDGDS	System dump (global)
SYSDUMPCODE(name)	DFHSDRDS	System dump (specific)
STORAGE(name)	DFHSMDDS	Storage manager domain subpool statistics (specific)
STORAGE	DFHSMSDS	Storage manager DSA (global)
TRANDUMPCODE	DFHTDGDS	Transaction dump (global)
TRANDUMPCODE(name)	DFHTDRDS	Transaction dump (specific)
TRANCLASS(name)	DFHXMCDS	Transaction class statistics (specific)
TRANSACTION	DFHXMGDS	Transaction manager statistics (global)
TRANSACTION(name)	DFHXMRDS	Transaction statistics (specific)
VTAM	DFHA03DS	VTAM statistics (global)

Note: Some of the above copybooks contain packed fields. Before using these fields, they should be checked for binary zeros. The COBOL versions of the fields have been redefined as numeric with a suffix of -R for this purpose.

COLLECT STATISTICS conditions

Condition	RESP2	Meaning
INVREQ	4	The LSRPOOL value is not in the range 1-8
IOERR	3	The requested statistics area was not functioning
NOTAUTH	100	You are not authorized to use this command
NOTAUTH	101	You are not authorized to access a specified resource
NOTFND	1	The named resource cannot be found
NOTFND	2	The type of resource is not defined to CICS (FEPI=NO in the SIT)

EXEC CICS PERFORM STATISTICS RECORD

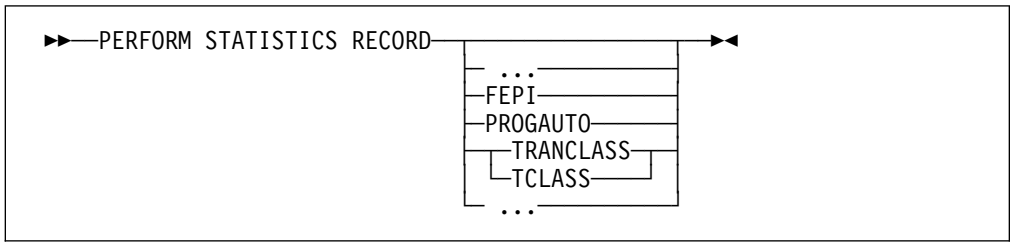
The following resource types are added for this release:

- FEPI to record FEPI statistics
- PROGAUTO to record program manager autoinstall statistics
- TRANCLASS to record transaction class statistics.

Function

The PERFORM STATISTICS RECORD command causes the statistics for a named resource to be written immediately to SMF.

Syntax



PERFORM STATISTICS options

FEPI

Resource statistics for FEPI.

PROGAUTO

Resource statistics for autoinstalled programs.

TRANCLASS

Resource statistics for the transaction class.

PERFORM STATISTICS conditions

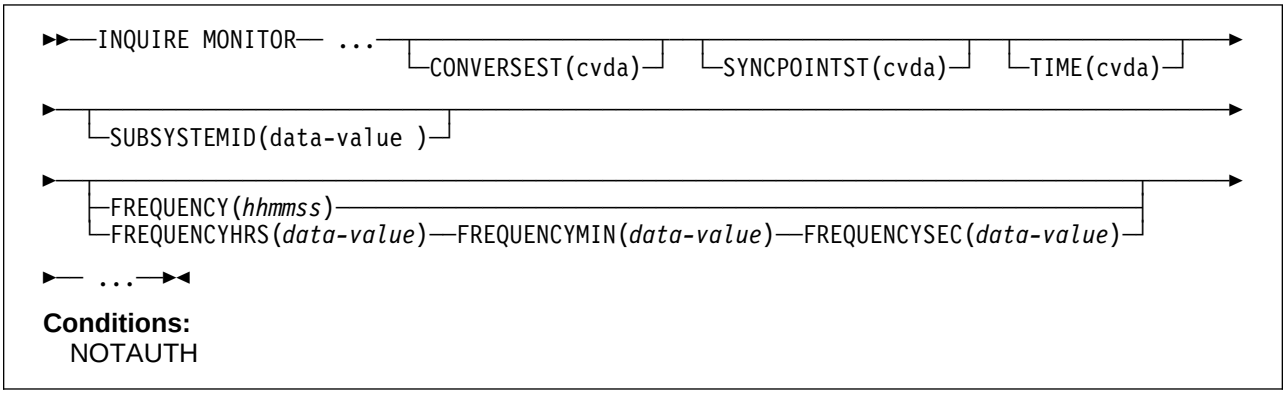
Condition	RESP2	Meaning
IOERR	n	At least one of these statistics areas was not functioning. The RESP2 value of “n” identifies the last unavailable type.
NOTAUTH	100	You are not authorized to use this command.
NOTFND	n	At least one of the resource types specified is not available. The RESP2 value of “n” identifies the last type not found.

EXEC CICS INQUIRE MONITOR command

Function

The INQUIRE MONITOR command allows you to find out whether CICS is accumulating monitoring data for executing transactions, and to discover which monitoring data classes are active.

Syntax



INQUIRE MONITOR options

CONVERSEST(cvda)

returns a CVDA value indicating whether conversational tasks are to have separate performance class records produced for each pair of terminal control I/O requests. (Converse or send/receive pair.) CVDA values are:

CONVERSE
: Conversational tasks are to have separate performance class records produced for each pair of terminal control I/O requests.

NOCONVERSE
: Conversational tasks are not to have separate performance class records.

FREQUENCY(data-area)

returns a full-word packed decimal value identifying the interval for which monitoring automatically produces a transaction performance class record for any long-running transaction that has been in the system for a time greater than the interval. FREQUENCY values can be either 000000 (meaning that FREQUENCY monitoring is inactive), or in the time range 15 minutes (001500) up to 24 hours (240000).

You can also specify a frequency using the separate FREQUENCYHRS, FREQUENCYMINS, and FREQUENCYSECS options.

If you ask for —HRS, —MINS, and —SECS, you must ask for all three—FREQUENCYHRS, FREQUENCYMINS, and FREQUENCYSECS must all be used to get the correct interval time.

FREQUENCYHRS(data-value)

returns a fullword binary field indicating the number of hours in the interval for which monitoring automatically produces a transaction performance class record for any long-running transaction. FREQUENCY values for hours are in the range 00–24.

FREQUENCYMIN(data-value)
returns a fullword binary field indicating the number of minutes in the interval for which monitoring automatically produces a transaction performance class record for any long-running transaction. FREQUENCY values for minutes are in the range 00–59.

FREQUENCYSEC(data-value)
returns a fullword binary field indicating the number of seconds in the interval for which monitoring automatically produces a transaction performance class record for any long-running transaction. FREQUENCY values for seconds are in the range 00–59.

SUBSYSTEMID(data-value)
returns a 4-character name used as the subsystem identification in the monitoring SYSEVENT class records. The subsystem identification defaults to the first four characters of the VTAM generic APPLID if MNSUBSYS is not specified as a system initialization parameter.

To enable transaction activity reporting with the Resource Measuring Facility (RMF), this value should correspond to the SUBSYS parameter in the IEAICSxx member of SYS1.PARMLIB. For more information about the use of the MVS IEAICS member see the *CICS/ESA Performance Guide*.

SYNCPOINTST(cvda)
returns a CVDA indicating whether a transaction performance class record is produced when a transaction takes an explicit or implicit syncpoint for a unit of work, except when the syncpoint is part of task termination or a syncpoint rollback.

CVDA values are:

NOSYNCPOINT
A transaction performance class record is not produced.

SYNCPOINT
A transaction performance class record is produced.

TIME(cvda)
returns a CVDA indicating whether the time stamp fields for in the performance class monitoring data are returned to an application in GMT or local time in response to an EXEC CICS COLLECT STATISTICS command.

CVDA values are:

GMT
The time stamp fields in the performance class monitoring data are Greenwich mean time (GMT).

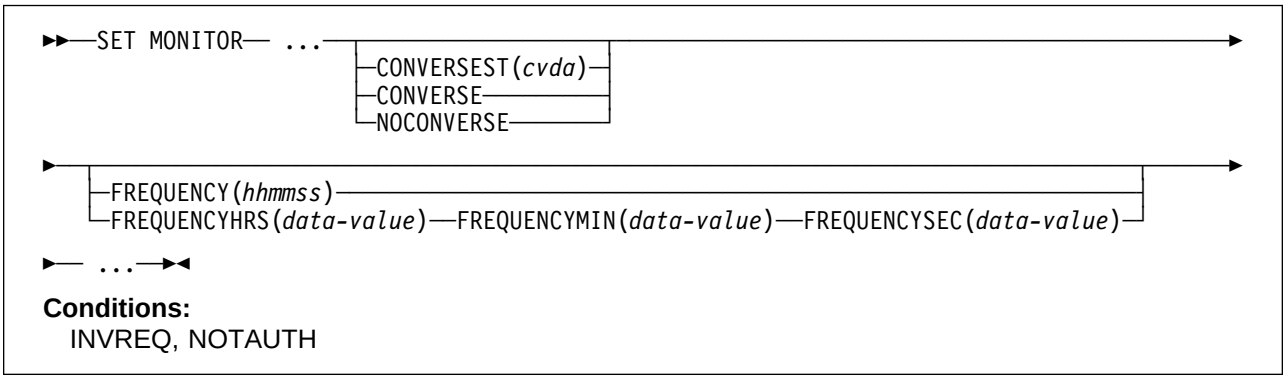
LOCAL
The time stamp fields in the performance class monitoring data are local time.

EXEC CICS SET MONITOR

Function

The SET MONITOR command allows you to switch CICS monitoring on or off and to select the classes of monitoring data to be collected.

Syntax



There are two formats for specifying a monitoring frequency:

- A composite time in packed decimal format (0hhmmss+) on the FREQUENCY parameter.
- Separate hours, minutes, and seconds intervals on the FREQUENCYHRS, FREQUENCYMIN, and FREQUENCYSEC parameters.

SET MONITOR options

CONVERSEST(cvda)
specifies a CVDA value to indicate whether conversational tasks are to have separate performance class records produced for each pair of terminal control I/O requests. (Converse or send/receive pair.) CVDA values are:

CONVERSE
Separate performance class records are to be produced.

NOCONVERSE
Separate performance class records are not to be produced.

FREQUENCY(data-value)
specifies, as a 4-byte packed decimal variable (0hhmmss+), the interval for which monitoring automatically produces a transaction performance class record for any long-running transaction that has been in the system for a timer greater than the interval.

FREQUENCYHRS
FREQUENCYMIN
FREQUENCYSEC
specify, as binary values, the frequency-setting parameters. When only one of these three fields is specified, the ranges allowed are:

FREQUENCYHRS 0–24
FREQUENCYMIN 0, or 15–59
FREQUENCYSEC 0, or 900–86400

When more than one of these fields is specified, the ranges allowed are:

FREQUENCYHRS 1–24
FREQUENCYMIN 0–59
FREQUENCYSEC 0–59

and the combined values must result in a zero value or a time in the range 15 minutes to 24 hours.

SET MONITOR conditions

Condition	*RESP2*	*Meaning*
INVREQ	5	CONVERSEST has an invalid CVDA value.
INVREQ	6	SYNCPOINTST has an invalid CVDA value.
INVREQ	7	FREQUENCY has an invalid value. • Hours ≤ 24, Mins ≤ 59, Secs ≤ 59 and (≥ 001500 ≤ 24000) • All 3 fields together, for example, 235959 is valid but 240001 is not.
INVREQ	8	Invalid Hours (00–24)
INVREQ	9	Invalid Minutes (00–59, or on its own 0, or 15–1 440)
INVREQ	10	Invalid Seconds (00–59, or on its own 0, 900–86 400)
NOTAUTH	100	The use of this command is not authorized.

End of General-use programming interface

Changes to CICS-supplied transactions

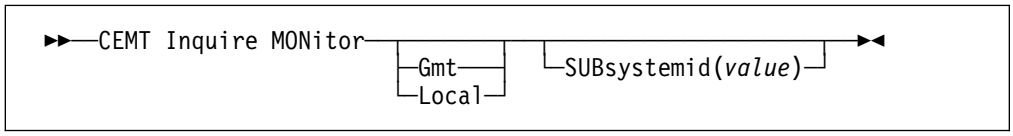
CEMT INQUIRE and SET MONITOR

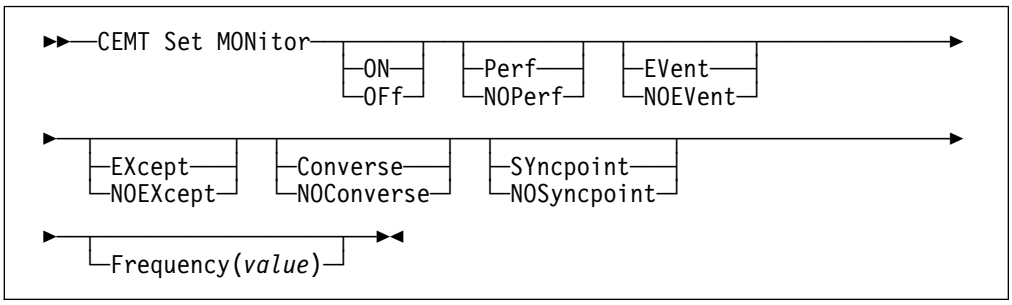
The CONVERSE, SYNCPOINT and FREQUENCY options are added to the CEMT INQUIRE and SET MONITOR commands.

Function: CEMT INQUIRE MONITOR returns the current settings of monitoring. In CICS/ESA 4.1, the subsystem id is returned, which is either the name you specify on the MNSUBSYS system initialization parameter or the first 4 characters of the APPLID. In addition time is reported as either GMT or Local. For example,

```
Mon On   Per    Exc          Fre( 000000 )  Gmt  Sub(IYAH)
```

Command syntax





Function: SET MONITOR allows the user to modify the following monitoring class settings and operational values.

Command options

CONVERSE
specifies that conversational tasks are to have separate performance class records produced for each pair of terminal control I/O requests.

FREQUENCY (value)
is the interval for which monitoring produces automatically a transaction performance class record for any long-running transaction. Frequency times are 0, or in the range 001500–24000. The default frequency value is 0 which means that FREQUENCY monitoring is inactive.

NOCONVERSE
specifies that conversational tasks are not to have separate performance class records produced for each pair of terminal control I/O requests.

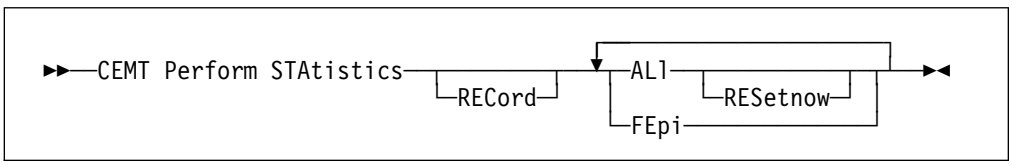
NOSYNCPOINT
specifies that a transaction performance class record is not to be produced when a transaction takes an explicit or implicit syncpoint (unit-of-work).

SYNCPOINT
specifies that a transaction performance class record is produced when a transaction takes an explicit or implicit syncpoint (unit-of-work), except when the syncpoint is part of task termination.

CEMT PERFORM STATISTICS RECORD

The FEPI option is added to the CEMT PERFORM STATISTICS RECORD command.

Command syntax



Command option

FEPI
FEPI statistics are to be written immediately to the SMF data set.

Changed sample programs

There are changes to the sample monitoring and statistics sample programs, DFH$MOLS and DFH0STAT.

DFH$MOLS sample monitoring program

DFH$MOLS is enhanced as follows for CICS/ESA 4.1:

- It formats and prints the additional SMF correlation data (JOBNAME and so on).
- It can handle CICS/ESA Version 3 and Version 4 SMF data records.
- It prints the new data from the exception data record.
- It no longer filters dictionary records on time.

DFH0STAT sample statistics program

DFH0STAT is enhanced as follows for CICS/ESA 4.1:

- The statistics sample program produces the following additional reports:
  - Monitoring and statistics status
  - Transaction manager
  - Program storage usage
  - Transactions and transaction totals
  - DFHRPL concatenation program analysis
  - LSR pools
  - Files and data tables.
- The statistics in the dispatcher report, the storage reports (above and below 16MB), and the loader report are significantly enhanced.
- You can now invoke DFH0STAT from a start-up, or shut-down, PLT program, and from the console.

Changed utility programs

- DFHSTUP now enables you to select reports by resource type, using two new control parameters. You can use these to select or ignore specific types of resource in your DFHSTUP report:

  SELECT TYPE
  IGNORE TYPE

  For more information about these parameters, see “Control parameters of the DFHSTUP program” on page 559.

DFHMNDUP monitoring utility program

DFHMNDUP is enhanced in CICS/ESA 4.1 to provide new operands for the extended SMF 110 product section.

Changed monitoring

The following sections describe the changes to monitoring data fields for CICS/ESA 4.1.

Additional performance class data fields

Product-sensitive programming interface

The following table shows the additional performance class data fields that are introduced in this release.

Table 23. Additional performance class data fields

Group Name	Field-Id	Description
DFHCICS	130	Transaction routing sysid
DFHFEPI	150	FEPI Allocate count
DFHFEPI	151	FEPI Receive count
DFHFEPI	152	FEPI Send count
DFHFEPI	153	FEPI Start count
DFHFEPI	154	FEPI CHARS sent
DFHFEPI	155	FEPI CHARS receive
DFHFEPI	156	FEPI Suspend time
DFHFEPI	157	FEPI Allocate time-out count
DFHFEPI	158	FEPI Receive time-out count
DFHFEPI	159	FEPI Total count
DFHSTOR	160	Program-storage high-water-mark (SDSA)
DFHSTOR	161	Program-storage high-water-mark (ESDSA)
DFHSTOR	162	Program-storage high-water-mark (RDSA)
DFHTASK	125	First dispatch delay time
DFHTASK	126	First dispatch delay time due to TRANCLASS
DFHTASK	127	First dispatch delay time due to MXT
DFHTASK	129	Task ENQ delay time
DFHTASK	166	Transaction class name
DFHTASK	170	Resource Manager Interface (RMI)–elapsed time
DFHTASK	170	Resource Manager Interface (RMI)–suspended time
DFHTERM	133	TC I/O wait time - LU6.1
DFHTERM	134	TC I/O wait time - LU6.2
DFHTERM	135	TC alternate facility input messages - LU6.2
DFHTERM	136	TC alternate facility output messages - LU6.2
DFHTERM	137	TC alternate facility CHARS input - LU6.2
DFHTERM	138	TC alternate facility CHARS output - LU6.2

Deleted performance class data fields

The following table shows the performance class data fields that are removed in this release.

Table 24. Deleted performance class data fields

Group Name	Field-Id	Description
DFHCICS	3	Operator id
DFHSTOR	104	Terminal control storage
DFHSTOR	140	Program-storage high-water-mark (EUDSA)
DFHSTOR	141	Program-storage high-water-mark (UDSA)
DFHTASK	110	Transaction class

Selectivity of performance class data fields

Selectivity of the CICS-data collected by monitoring can be controlled by DFHMCT parameters. These (EXCLUDE= and INCLUDE=) apply to the TYPE=RECORD parameter for performance class monitoring. Each parameter may specify one or more fields, either specifically by field-id, or generically by group-name. The EXCLUDE parameter will be honored before any INCLUDE parameter. A revised list of field-ids and group-names that are eligible for exclusion or inclusion follows.

Note: Not all fields can be selected in this way; only those listed here.

Table 25 (Page 1 of 3). This table shows the data groups and fields that can be excluded/included

Group Name	Field Id	Description
DFHCICS	103	Transaction exception wait time
DFHCICS	112	Performance record type
DFHCICS	130	Transaction routing sysid
DFHDEST	41	TD get count
DFHDEST	42	TD put count
DFHDEST	43	TD purge count
DFHDEST	91	TD total count
DFHDEST	101	TD I/O wait time
DFHFEPI	150	FEPI Allocate count
DFHFEPI	151	FEPI Receive count
DFHFEPI	152	FEPI Send count
DFHFEPI	153	FEPI Start count
DFHFEPI	154	FEPI CHARS sent
DFHFEPI	155	FEPI CHARS received
DFHFEPI	156	FEPI Suspend time
DFHFEPI	157	FEPI Allocate time-out count
DFHFEPI	158	FEPI Receive time-out count
DFHFEPI	159	FEPI Total count
DFHFILE	36	FC get count
DFHFILE	37	FC put count
DFHFILE	38	FC browse count
DFHFILE	39	FC add count
DFHFILE	40	FC delete count
DFHFILE	63	FC I/O wait time
DFHFILE	70	FC access-method count
DFHFILE	93	FC total count
DFHJOUR	10	JC I/O wait time

Table 25 (Page 2 of 3). This table shows the data groups and fields that can be excluded/included

Group Name	Field Id	Description
DFHJOUR	58	JC put/write count
DFHMAPP	50	BMS MAP count
DFHMAPP	51	BMS IN count
DFHMAPP	52	BMS OUT count
DFHMAPP	90	BMS total count
DFHPROG	55	Program LINK count
DFHPROG	56	Program XCTL count
DFHPROG	57	Program LOAD count
DFHPROG	71	Program name
DFHPROG	113	Original abend code
DFHPROG	114	Current abend code
DFHPROG	115	Program load time
DFHSTOR	33	User-storage high-water-mark (UDSA)
DFHSTOR	54	User-storage get-count (UDSA)
DFHSTOR	87	Program-storage high-water-mark - total
DFHSTOR	95	User-storage-occupancy (bytes-ms) (UDSA)
DFHSTOR	105	User-storage get-count–above 16MB (EUDSA)
DFHSTOR	106	User-storage high-water-mark–above 16MB (EUDSA)
DFHSTOR	107	User-storage-occupancy (bytes-ms)–above 16MB (EUDSA)
DFHSTOR	108	Program-storage high-water-mark–below 16MB
DFHSTOR	116	User-storage high-water-mark–below 16MB (CDSA)
DFHSTOR	117	User-storage get-count–below 16MB (CDSA)
DFHSTOR	118	User-storage-occupancy (bytes-ms)–below 16MB (CDSA)
DFHSTOR	119	User-storage high-water-mark–above 16MB (ECDSA)
DFHSTOR	120	User-storage get-count–above 16MB (ECDSA)
DFHSTOR	121	User-storage-occupancy (bytes-ms)–above 16MB (ECDSA)
DFHSTOR	122	Program-storage high-water-mark (ERDSA)
DFHSTOR	139	Program-storage high-water-mark–above 16MB
DFHSTOR	142	Program-storage high-water-mark (ECDSA)
DFHSTOR	143	Program-storage high-water-mark (CDSA)
DFHSTOR	160	Program-storage high-water-mark (SDSA)
DFHSTOR	161	Program-storage high-water-mark (ESDSA)
DFHSTOR	162	Program-storage high-water-mark (RDSA)
DFHSYNC	60	Sync point count
DFHTASK	7	User-task dispatch time
DFHTASK	8	User-task CPU time
DFHTASK	14	User-task suspend time
DFHTASK	31	Task number
DFHTASK	59	IC put/initiate count
DFHTASK	64	Error flag field
DFHTASK	97	Network name of the originating terminal or system
DFHTASK	98	Unit-of-work id on the originating system
DFHTASK	102	User-task wait-for-dispatch time
DFHTASK	109	Transaction priority
DFHTASK	125	First dispatch delay time
DFHTASK	126	First dispatch delay time due to TRANCLASS

Table 25 (Page 3 of 3). This table shows the data groups and fields that can be excluded/included

Group Name	Field Id	Description
DFHTASK	127	First dispatch delay due to MXT
DFHTASK	129	Task ENQ delay time
DFHTASK	170	Resource Manager Interface–elapsed time
DFHTASK	171	Resource Manager Interface–suspended time
DFHTEMP	11	TS I/O wait time
DFHTEMP	44	TS get count
DFHTEMP	46	TS put auxiliary count
DFHTEMP	47	TS put main count
DFHTEMP	92	TS total count
DFHTERM	9	TC I/O wait time
DFHTERM	34	TC principal facility input messages
DFHTERM	35	TC principal facility output messages
DFHTERM	67	TC alternate facility input messages
DFHTERM	68	TC alternate facility output messages
DFHTERM	69	TC allocate count
DFHTERM	83	TC principal facility CHARS input
DFHTERM	84	TC principal facility CHARS output
DFHTERM	85	TC alternate facility CHARS input
DFHTERM	86	TC alternate facility CHARS output
DFHTERM	100	IR I/O wait time
DFHTERM	111	VTAM terminal LU name
DFHTERM	133	TC I/O wait time - LU6.1
DFHTERM	134	TC I/O wait time - LU6.2
DFHTERM	135	TC alternate facility input messages - LU6.2
DFHTERM	136	TC alternate facility output messages - LU6.2
DFHTERM	137	TC alternate facility CHARS input - LU6.2
DFHTERM	138	TC alternate facility CHARS output - LU6.2

Performance data in group DFHCICS

005 (TYPE-T, 'START', 8 BYTES)

Start time of measurement interval. This is either the time at which the user task was attached, or the time at which data recording was most recently reset in support of the MCT user event monitoring point DELIVER option, or the monitoring options MNSYNC, MNCONV, or FREQUENCY.

006 (TYPE-T, 'STOP', 8 BYTES)

Finish time of measurement interval. This is either the time at which the user task was detached, or the time at which data recording was completed in support of the MCT user event monitoring point DELIVER option or the monitoring options MNSYNC, MNCONV, or FREQUENCY.

Note: Response Time = STOP – START.

089 (TYPE-C, 'USERID', 8 BYTES)

User identification at task creation. This can also be the remote user identifier for a task created as the result of receiving an ATTACH request across an MRO or APPC link with attach-time security enabled.

103 (TYPE-S, 'EXWTTIME', 8 BYTES)
Accumulated data for exception conditions. The 32-bit clock contains the total elapsed time for which the user waited on exception conditions. The 24-bit period count equals the number of exception conditions that have occurred for this task.

Note: The performance class data field 'exception wait time' will be updated when exception conditions are encountered even when the exception class is inactive.

112 (TYPE-C, 'RTYPE', 4 BYTES)
Performance record type (low-order byte-3):

C Record output for a terminal converse
D Record output for a user EMP DELIVER request
F Record output for a long-running transaction
S Record output for a syncpoint
T Record output for a task termination.

130 (TYPE-C, 'RSYSID', 4 bytes)
is the name (sysid) of the remote system to which this transaction was routed either statically or dynamically.

Note: If the transaction was not routed or was routed locally, this field is set to null. Also see the program name (field 71).

Performance data in group DFHFEPI

150 (TYPE-A,'SZALLOCT', 4 bytes)
Number of conversations allocated by the user task. This number is incremented for each FEPI ALLOCATE POOL or FEPI CONVERSE POOL.

151 (TYPE-A,'SZRCVCT',4 bytes)
Number of FEPI RECEIVE requests made by the user task. This number is also incremented for each FEPI CONVERSE request.

152 (TYPE-A,'SZSENDCT',4 bytes)
Number of FEPI SEND requests made by the user task. This number is also incremented for each FEPI CONVERSE request.

153 (TYPE-A,'SZSTRTCT',4 bytes)
Number of FEPI START requests made by the user task.

154 (TYPE-A,'SZCHROUT',4 bytes)
Number of characters sent through FEPI by the user task.

155 (TYPE-A,'SZCHRIN',4 bytes)
Number of characters received through FEPI by the user task.

156 (TYPE-S,'SZWAIT',8 bytes)
Elapsed time in which the user task waited for all FEPI services.

157 (TYPE-A,'SZALLCTO',4 bytes)
Number of times the user task timed out while waiting to allocate a conversation.

158 (TYPE-A,'SZRCVTO',4 bytes)
Number of times the user task timed out while waiting to receive data.

159 (TYPE-A,'SZTOTCT',4 bytes)
Total number of all FEPI API and SPI requests made by the user task.

Performance data in group DFHSTOR

User storage fields in group DFHSTOR:

033 (TYPE-A, 'SCUSRHWM', 4 BYTES)
Maximum amount (high-water mark) of user storage allocated to the user task below 16MB, in the user dynamic storage area (UDSA).

054 (TYPE-A, 'SCUGETCT', 4 BYTES)
Number of user-storage GETMAIN requests issued by the user task below 16MB in the UDSA.

095 (TYPE-A, 'SCUSRSTG', 8 BYTES)
Storage occupancy of the user task below 16MB, in the UDSA. This measures the area under the curve of storage in use against elapsed time.

105 (TYPE-A, 'SCUGETCT', 4 BYTES)
Number of user-storage GETMAIN requests issued by the user task for storage above 16MB, in the extended user dynamic storage area (EUDSA).

106 (TYPE-A, 'SCUSRHWM', 4 BYTES)
Maximum amount (high-water mark) of user-storage allocated to the user task above 16MB, in the EUDSA.

107 (TYPE-A, 'SCUCRSTG', 8 BYTES)
Storage occupancy of the user task above 16MB, in the EUDSA. This measures the area under the curve of storage in use against elapsed time.

116 (TYPE-A, 'SC24CHWM', 4 BYTES)
Maximum amount (high-water mark) of user-storage allocated to the user task below 16MB, in the CICS dynamic storage area (CDSA).

117 (TYPE-A, 'SCCGETCT', 4 BYTES)
Number of user-storage GETMAIN requests issued by the user task for storage below 16MB, in the CDSA.

118 (TYPE-A, 'SC24COCC', 8 BYTES)
Storage occupancy of the user task below 16MB, in the CDSA. This measures the area under the curve of storage in use against elapsed time.

119 (TYPE-A, 'SC31CHWM', 4 BYTES)
Maximum amount (high-water mark) of user-storage allocated to the user task above 16MB, in the extended CICS dynamic storage area (ECDSA).

120 (TYPE-A, 'SCCGETCT', 4 BYTES)
Number of user-storage GETMAIN requests issued by the user task for storage above 16MB, in the ECDSA.

121 (TYPE-A, 'SC31COCC', 8 BYTES)
Storage occupancy of the user task above 16MB, in the ECDSA. This measures the area under the curve of storage in use against elapsed time.

Table 26. User storage field id cross reference				
	UDSA	**EUDSA**	**CDSA**	**ECDSA**
Getmain count	054	105	117	120
High-water-mark	033	106	116	119
Occupancy	095	107	118	121

Program storage fields in group DFHSTOR

087 (TYPE-A, 'PCSTGHWM', 4 BYTES)
Maximum amount (high-water mark) of program storage in use by the user task both above **and** below 16MB.

108 (TYPE-A, 'PC24BHWM', 4 BYTES)
Maximum amount (high-water mark) of program storage in use by the user task below 16MB. This field is a subset of PCSTGHWM (field ID 087) that resides below 16MB.

122 (TYPE-A, 'PC31RHWM', 4 BYTES)
Maximum amount (high-water mark) of program storage in use by the user task above 16MB, in the extended read-only dynamic storage area (ERDSA). This field is a subset of PC31AHWM (field ID 139) that resides in the ERDSA.

139 (TYPE-A, 'PC31AHWM', 4 BYTES)
Maximum amount (high-water mark) of program storage in use by the user task above 16MB. This field is a subset of PCSTGHWM (field ID 087) that resides above 16MB.

142 (TYPE-A, 'PC31CHWM', 4 BYTES)
Maximum amount (high-water mark) of program storage in use by the user task above 16MB, in the extended CICS dynamic storage area (ECDSA). This field is a subset of PC31AHWM (139) that resides in the ECDSA.

143 (TYPE-A, 'PC24CHWM', 4 BYTES)
Maximum amount (high-water mark) of program storage in use by the user task below 16MB, in the CICS dynamic storage area (CDSA). This field is a subset of PC24BHWM (108) that resides in the CDSA.

160 (TYPE-A, 'PC24SHWM', 4 BYTES)
Maximum amount (high-water mark) of program storage in use by the user task below 16MB, in the shared dynamic storage area (SDSA). This field is a subset of PC24BHWM (108) that resides in the SDSA.

161 (TYPE-A, 'PC31SHWM', 4 BYTES)
Maximum amount (high-water mark) of program storage in use by the user task above 16MB, in the extended shared dynamic storage area (ESDSA). This field is a subset of PC31AHWM (139) that resides in the ESDSA.

162 (TYPE-A, 'PC24RHWM', 4 BYTES)
Maximum amount (high-water mark) of program storage in use by the user task below 16MB, in the read-only dynamic storage area (RDSA). This field is a subset of PC24BHWM (108) that resides in the RDSA.

Performance data in group DFHTASK

001 (TYPE-C, 'TRAN', 4 BYTES)
Transaction identification.

004 (TYPE-C, 'T', 4 BYTES)
Transaction start type. The high-order bytes (0 and 1) are set to:

TO Attached from terminal input
S Attached by automatic transaction initiation (ATI) without data
SD Attached by automatic transaction initiation (ATI) with data
QD Attached by transient data trigger level
U Attached by user request
TP Attached from terminal TCTTE transaction ID
SZ Attached by the front-end programming interface (FEPI).

007 (TYPE-S, 'USRDISPT', 8 BYTES)
Elapsed time for which the user task was dispatched.

008 (TYPE-S, 'USRCPUT', 8 BYTES)
Processor time for which the user task was dispatched on each CICS TCB (QR, RO, CO).

014 (TYPE-S, 'SUSPTIME', 8 BYTES)
Total elapsed wait time for which the user task was suspended by the dispatcher. This includes:

- The elapsed time waiting for the first dispatch. This also includes any delay incurred because of the limits set for this transaction's transaction class (if any) or by the system parameter MXT being reached.
- The task suspend (wait) time.
- The elapsed time waiting for re-dispatch after a suspended task has been resumed.

031 (TYPE-P, 'TRANNUM', 4 BYTES)
Transaction identification number.

Note: The transaction number field is normally a 4-byte packed decimal number. However, some CICS system tasks are identified by special character 'transaction numbers', as follows:

- ' III' for system initialization task
- ' JBS' or ' Jnn' for journal control (where nn = the journal number from 01–99
- ' TCP' for terminal control.

These special identifiers are placed in bytes 2 through 4. Byte 1 is a blank (X'40') before the terminal control TCP identifier, and a null value (X'00') before the others.

059 (TYPE-A, 'ICPUINCT', 4 BYTES)
Number of interval control START or INITIATE requests during the user task.

064 (TYPE-A, 'TASKFLAG', 4 BYTES)
Task error flags, a string of 31 bits used for signaling unusual conditions occurring during the user task:

Bit 0 Reserved

Bit 1 Detected an attempt either to start a user clock that was already running, or to stop one that was not running

Bits 2–31 Reserved.

097 (TYPE-C, 'NETNAME', 20 BYTES)

Fully qualified name by which the originating system is known to the VTAM network. This name is assigned at attach time using either the NETNAME derived from the TCT (when the task is attached to a local terminal), or the NETNAME passed as part of an ISC APPC or IRC attach header. At least three padding bytes (X'00') are present at the right end of the name.

If the originating terminal is VTAM across an ISC APPC or IRC link, then the NETNAME is the *networkid.LUname* If the terminal is non-VTAM, then the NETNAME is *networkid.generic_applid*.

All originating information passed as part of an ISC LUTYPE6.1 attach header has the same format as the non-VTAM terminal originators above.

When the originator is communicating over a DL/I batch session, the name is a concatenation of 'jobname.stepname.procname' derived from the originating system. Characters in excess of 17 cause truncation **at the left**.

When the originator is communicating over an external CICS interface (EXCI) session, the name is a concatenation of the following:

```
'DFHEXCI │ TCB address │ Address Space Id  │ MVS Id'
 7 bytes│ 4 bytes     │ 2 bytes           │ 4 bytes
```

These components are derived from the originating system.

098 (TYPE-C, 'UOWID', 8 BYTES)

Name by which the unit of work is known within the originating system. This name is assigned at attach time using either an STCK-derived token (when the task is attached to a local terminal), or the unit of work ID passed as part of an ISC APPC or IRC attach header.

The first six bytes of this field are one of the following:

- A binary value derived from the clock of the originating system and wrapping round at intervals of several months
- A character value of the form "hhmmss", which wraps round daily. This case applies when the originating system is communicating through a DL/I batch session.

The last two bytes of this field are for the period count. These may change during the life of the task as a result of syncpoint activity.

Note: When using MRO or ISC, the UOWID field must be combined with the NETNAME field (097) to uniquely identify a task, because UOWID is unique only to the originating CICS system.

102 (TYPE-S, 'DISPWTT', 8 BYTES)

Elapsed time for which the user task waited for re-dispatch. This is the aggregate of the wait times between each event completion and user-task re-dispatch.

Note: This field does not include the elapsed time spent waiting for first dispatch. This field is a subset of the task suspend time, SUSPTIME (014), field.

109 (TYPE-C, 'TRANPRI', 4 BYTES)
Transaction priority when monitoring of the task was initialized (low-order byte-3).

125 (TYPE-S, 'DSPDELAY', 8 BYTES)
The elapsed time waiting for first dispatch.

Note: This field is a subset of the task suspend time, SUSPTIME (014), field.

126 (TYPE-S, 'TCLDELAY', 8 BYTES)
The elapsed time waiting for first dispatch which was delayed because of the limits set for this transaction's transaction class, TCLSNAME (166), being reached.

Note: This field is a subset of the first dispatch delay, DSPDELAY (125), field.

127 (TYPE-S, 'MXTDELAY', 8 BYTES)
The elapsed time waiting for first dispatch which was delayed because of the limits set by the system parameter, MXT, being reached.

Note: The field is a subset of the first dispatch delay, DSPDELAY (125), field.

129 (TYPE-S, 'ENQDELAY', 8 BYTES)
The elapsed time waiting for a CICS Task Control ENQ.

Note: This field is a subset of the task suspend time, SUSPTIME (014), field.

166 (TYPE-C, 'TCLSNAME', 8 BYTES)
Transaction class name. This field is null if the transaction is not in a TRANCLASS.

170 (TYPE-S, 'RMITIME', 8 BYTES)
Amount of elapsed time spent in the Resource Manager Interface (RMI).

171 (TYPE-S, 'RMISUSP', 8 BYTES)
Amount of elapsed time the task was suspended by the dispatcher while in the Resource Manager Interface (RMI).

Note: The field is a subset of the task suspend time, SUSPTIME (014), field and also the RMITIME (170) field.

Performance data in group DFHTERM

002 (TYPE-C, 'TERM', 4 BYTES)
Terminal or session identification. This field is null if the task is not associated with a terminal or session.

009 (TYPE-S, 'TCIOWTT', 8 BYTES)
Elapsed time for which the user task waited for input from the terminal operator, after issuing a RECEIVE request.

034 (TYPE-A, 'TCMSGIN1', 4 BYTES)
Number of messages received from the task's principal terminal facility, including LUTYPE6.1 and LUTYPE6.2 (APPC) but not MRO (IRC).

035 (TYPE-A, 'TCMSGOU1', 4 BYTES)
Number of messages sent to the task's principal terminal facility, including LUTYPE6.1 and LUTYPE6.2 (APPC) but not MRO (IRC).

067 (TYPE-A, 'TCMSGIN2', 4 BYTES)
Number of messages received from the LUTYPE6.1 alternate terminal facilities by the user task.

068 (TYPE-A, 'TCMSGOU2', 4 BYTES)
Number of messages sent to the LUTYPE6.1 alternate terminal facilities by the user task.

069 (TYPE-A, 'TCALLOCT', 4 BYTES)
Number of TCTTE ALLOCATE requests issued by the user task for LUTYPE6.2 (APPC), LUTYPE6.1, and IRC sessions.

083 (TYPE-A, 'TCCHRIN1', 4 BYTES)
Number of characters received from the task's principal terminal facility, including LUTYPE6.1 and LUTYPE6.2 (APPC) but not MRO (IRC).

084 (TYPE-A, 'TCCHROU1', 4 BYTES)
Number of characters sent to the task's principal terminal facility, including LUTYPE6.1 and LUTYPE6.2 (APPC) but not MRO (IRC).

085 (TYPE-A, 'TCCHRIN2', 4 BYTES)
Number of characters received from the LUTYPE6.1 alternate terminal facilities by the user task. *(Not applicable to ISC APPC.)*

086 (TYPE-A, 'TCCHROU2', 4 BYTES)
Number of characters sent to the LUTYPE6.1 alternate terminal facilities by the user task. *(Not applicable to ISC APPC.)*

100 (TYPE-S, 'IRIOWTT', 8 BYTES)
Elapsed time for which the user task waited for control at this end of an MRO link.

111 (TYPE-C, 'LUNAME', 8 BYTES)
VTAM logical unit name (if available) of the terminal associated with this transaction. If the task is executing in an application-owning or file-owning region then the luname is the generic applid of the originating connection for LUTYPE6.1 and LUTYPE6.2 (APPC). The luname is blank if the originating connection is either MRO (ISC) or external CICS interface (EXCI).

133 (TYPE-S, 'LU61WTT', 8 BYTES)
The elapsed time for which the user task waited for I/O on a LUTYPE6.1 connection or session. This time also includes the waits incurred for conversations across LUTYPE6.1 connections, but not the waits incurred due to LUTYPE6.1 syncpoint flows.

134 (TYPE-S, 'LU62WTT', 8 BYTES)
The elapsed time for which the user task waited for I/O on a LUTYPE6.2 (APPC) connection or session. This time also includes the waits incurred for conversations across LUTYPE6.2 (APPC) connections, but not the waits incurred due to LUTYPE6.2 (APPC) syncpoint flows.

135 (TYPE-A, 'TCM62IN2', 4 BYTES)
Number of messages received from the alternate facility by the user task for LUTYPE6.2 (APPC) sessions.

136 (TYPE-A, 'TCM62OU2', 4 BYTES)
Number of messages sent to the alternate facility by the user task for LUTYPE6.2 (APPC) sessions.

137 (TYPE-A, 'TCC62IN2', 4 BYTES)
Number of characters received from the alternate facility by the user task for LUTYPE6.2 (APPC) sessions.

138 (TYPE-A, 'TCC62OU2', 4 BYTES)
Number of characters sent to the alternate facility by the user task for LUTYPE6.2 (APPC) sessions.

Exception class data

Exception records are produced after each of the following conditions encountered by a transaction has been resolved:

- Wait for storage in the CDSA
- Wait for storage in the UDSA
- Wait for storage in the SDSA
- Wait for storage in the RDSA
- Wait for storage in the ECDSA
- Wait for storage in the EUDSA
- Wait for storage in the ESDSA
- Wait for storage in the ERDSA
- Wait for auxiliary temporary storage
- Wait for auxiliary temporary storage string
- Wait for auxiliary temporary storage buffer
- Wait for file string
- Wait for file buffer
- Wait for LSRPOOL string

These records are fixed format. For the format of these exception records is as follows:

```
MNEXCDS  DSECT
EXCMNTRN DS    CL4               TRANSACTION IDENTIFICATION
EXCMNTER DS    XL4               TERMINAL IDENTIFICATION
EXCMNUSR DS    CL8               USER IDENTIFICATION
EXCMNTST DS    CL4               TRANSACTION START TYPE
EXCMNSTA DS    XL8               EXCEPTION START TIME
EXCMNSTO DS    XL8               EXCEPTION STOP TIME
EXCMNTNO DS    PL4               TRANSACTION NUMBER
EXCMNTPR DS    XL4               TRANSACTION PRIORITY
         DS    CL4               RESERVED
EXCMNLUN DS    CL8               LUNAME
         DS    CL4               RESERVED
EXCMNEXN DS    XL4               EXCEPTION NUMBER
EXCMNRTY DS    CL8               EXCEPTION RESOURCE TYPE
EXCMNRID DS    CL8               EXCEPTION RESOURCE ID
EXCMNTYP DS    XL2               EXCEPTION TYPE
EXCMNWT  EQU   X'0001'             WAIT
EXCMNBWT EQU   X'0002'             BUFFER WAIT
EXCMNSWT EQU   X'0003'             STRING WAIT
         DS    CL2               RESERVED
EXCMNTCN DS    CL8               TRANSACTION CLASS NAME
*        END OF EXCEPTION RECORD ...
```

Exception data field descriptions

EXCMNTRN (TYPE-C, 4 BYTES)
Transaction identification. This field is null if the task is not associated with a terminal or session.

EXCMNTER (TYPE-C, 4 BYTES)
Terminal identification. This field is null if the task is not associated with a terminal or session.

EXCMNUSR (TYPE-C, 8 BYTES)
User identification at task creation. This can also be the remote user identifier for a task created as the result of receiving an ATTACH request across an MRO or APPC link with attach-time security enabled.

EXCMNTST (TYPE-C, 4 BYTES)
Transaction start type. The low-order byte (0 and 1) is set to:

- **'TO'** Attached from terminal input
- **'S'** Attached by automatic transaction initiation (ATI) without data
- **'SD'** Attached by automatic transaction initiation (ATI) with data
- **'QD'** Attached by transient data trigger level
- **'U '** Attached by user request
- **'TP'** Attached from terminal TCTTE transaction ID
- **'SZ'** Attached by Front End Programming Interface (FEPI)

EXCMNSTA (TYPE-T, 8 BYTES)
Start time of the exception.

EXCMNSTO (TYPE-T, 8 BYTES)
Finish time of the exception.

Note: The performance class exception wait time field, EXWTTIME (103), is a calculation based on subtracting the start time of the exception (EXCMNSTA) from the finish time of the exception (EXCMNSTO).

EXCMNTNO (TYPE-P, 4 BYTES)
Transaction identification number.

EXCMNTPR (TYPE-C, 4 BYTES)
Transaction priority when monitoring was initialized for the task (low-order byte).

EXCMNLUN (TYPE-C, 4 BYTES)
VTAM logical unit name (if available) of the terminal associated with this transaction. This field is nulls if the task is not associated with a terminal.

EXCMNEXN (TYPE-A, 4 BYTES)
Exception sequence number for this task.

EXCMNRTY (TYPE-C, 8 BYTES)
Exception resource type. The possible values for EXCMNRTY are shown in Table 27 on page 324.

EXCMNRID (TYPE-C, 8 BYTES)
Exception resource identification. The possible values for EXCMNRID are shown in Table 27 on page 324.

EXCMNTYP (TYPE-A, 2 BYTES)
Exception type. This field can be set to one of the following values:

X'0001' Exception due to a wait (EXCMNWT)
X'0002' Exception due to a buffer wait (EXCMNBWT)
X'0003' Exception due to a string wait (EXCMNSWT)

EXCMNTCN (TYPE-C, 8 BYTES)
Transaction class name. This field is null if the transaction is not in a transaction class.

The following table shows the value and relationships of the fields EXCMNTYP, EXCMNRTY, and EXCMNRID.

Table 27. Possible values of EXCMNTYP, EXCMNRTY, and EXCMNRID. The relationship between exception type, resource type, and resource identification.

EXCMNTYP Exception type	EXCMNRTY Resource type	EXCMNRID Resource ID	MEANING
EXCMNWT	'STORAGE'	'UDSA'	Wait for UDSA storage
EXCMNWT	'STORAGE'	'EUDSA'	Wait for EUDSA storage
EXCMNWT	'STORAGE'	'CDSA'	Wait for CDSA storage
EXCMNWT	'STORAGE'	'ECDSA'	Wait for ECDSA storage
EXCMNWT	'STORAGE'	'SDSA'	Wait for SDSA storage
EXCMNWT	'STORAGE'	'ESDSA'	Wait for ESDSA storage
EXCMNWT	'STORAGE'	'RDSA'	Wait for RDSA storage
EXCMNWT	'STORAGE'	'ERDSA'	Wait for ERDSA storage
EXCMNWT	'TEMPSTOR'	TS Qname	Wait for temporary storage
EXCMNSWT	'FILE'	filename	Wait for string associated with file
EXCMNSWT	'LSRPOOL'	filename	Wait for string associated with LSRPOOL
EXCMNSWT	'TEMPSTOR'	TS Qname	Wait for string associated with DFHTEMP
EXCMNBWT	'LSRPOOL'	LSRPOOL	Wait for buffer associated with LSRPOOL
EXCMNBWT	'TEMPSTOR'	TS Qname	Wait for buffer associated with DFHTEMP

End of Product-sensitive programming interface

Part 5. CICS internal restructure

This Part describes the major components of CICS that are restructured in CICS/ESA 4.1, and rewritten to conform to the CICS/ESA domain architecture. It also describes the restructure of the CICS terminal control signon component, which remains part of the AP domain. It includes the following topics:

Chapter 19. Directory manager domain

This chapter describes the directory manager domain introduced in CICS/ESA 4.1. It covers the following topics:

- Overview
- Benefits of the directory manager domain
- Changes to CICS externals
- Problem determination.

Overview

The directory manager domain provides resource-table lookup services for the CICS/ESA 3.3 components that are now restructured into new domains, such as transaction manager, program manager, and user domains. It replaces the services previously provided when these domains were components of the AP domain, and therefore able to use the CICS table manager program (DFHTMP). The table manager program is not available outside the AP domain, hence the new domains require the services of the directory manager domain to provide equivalent function.

Note that as a consequence of the restructure, four of the internal tables used in earlier releases of CICS no longer exist—the PCT, PPT, PCTR, and TPNT. The components that used these obsolete tables are part of the CICS/ESA 4.1 restructure, and now use the directory manager domain.

The resource definitions for which the directory manager domain provides the new services are as follows:

- Transaction definitions
- Remote transaction definitions
- Transaction classes
- TPNames
- User attributes
- Programs
- BMS mapsets
- BMS partition sets.

The AP domain continues to use the lookup services of DFHTMP for the other types of resource.

Benefits of the directory manager domain

The directory manager domain provides internal services to other domains, with improved performance compared with DFHTMP.

Like all the restructured components of CICS, it also provides higher reliability and serviceability.

Changes to CICS externals

The introduction of the directory manager domain results in a number of changes to CICS externals: These are:

- Changes to system definition
- Changes to the system programming interface (SPI)
- Changes to CICS-supplied transactions.

Changes to system definition

The following new system initialization parameters have been added to support the directory manager domain:

- SPCTRDD
- STNTRDD

These parameters are defined as for the other SPCTRxx system initialization parameters. They can be specified as PARM or SYSIN data, or as console input. These are shown in Table 28.

Table 28. The DFHSIT macro parameters

	DFHSIT	[TYPE={**CSECT**|DSECT}] ⋮ [SPCTRDD={(1[,2][,3])|ALL|OFF}] [STNTRDD={(1[,2][,3])|ALL|OFF}] ⋮
	END	DFHSITBA

SPCTRDD={(1[,2][,3])|ALL|OFF}
Sets the level of special tracing for the directory manager domain.

See the *CICS/ESA System Definition Guide* for information about specifying special trace levels using the SPCTRxx system initialization parameters.

STNTRDD={(1[,2][,3])|ALL|OFF}
Sets the level of standard tracing for the directory manager domain.

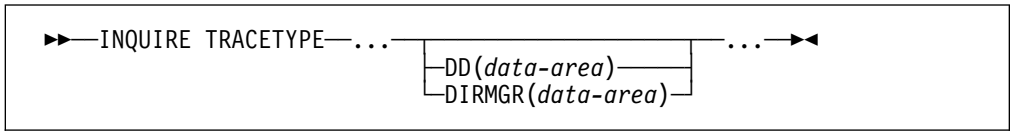
See the *CICS/ESA System Definition Guide* for information about specifying standard trace levels using the STNTRxx system initialization parameters.

Changes to the system programming interface

General-use programming interface

The DD and DIRMGR options are added to the INQUIRE and SET TRACETYPE commands.

INQUIRE TRACETYPE command

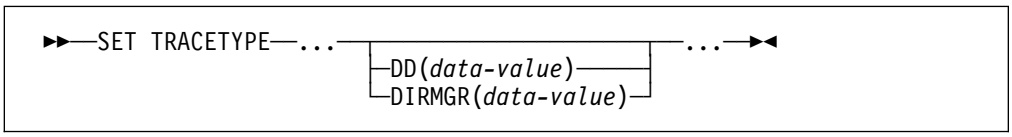


INQUIRE TRACETYPE options

DD(data-area) or DIRMGR(data-area)

returns the trace levels for the directory manager domain, in the form of a 32-bit string, where the bit positions correspond to the trace levels.

CICS returns either the standard or special flags, depending on the SPECIAL or STANDARD option specified on the command.



SET TRACETYPE options

DD(data-value) or DIRMGR(data-value)

sets the trace levels for the directory manager domain, in the form of a 32-bit string, where the bit positions correspond to the trace levels.

CICS sets either the standard or special flags, depending on the SPECIAL or STANDARD option specified on the command.

End of General-use programming interface

End of General-use programming interface

Changes to CICS-supplied transactions

The component keyword DD is added to the components display of the CETR transaction. You can use the CETR panel to view and modify the trace levels for the directory manager.

Problem determination

The following are provided to help with problem determination:

- Two new messages have been added
- A new section has been added to system dumps
- New trace points are added

New messages and codes

There are two new standard messages provided for the directory manager domain. They are:

DFHDD0001 for abends
DFHDD0002 for severe errors

For details of all the new, changed, and obsolete messages, see the *CICS/ESA Migration Guide*.

Changes to system dump

The DD keyword is added to the CICS-supplied IPCS dump exit for formatting the program manager domain state data. The following control blocks are formatted in the system dump.

- The DD anchor block
- Every directory
- Every header for each directory
- Every current browse for each directory
- Every collision list element for each directory.

Trace

The new trace entries for the directory manager domain are provided in the *CICS/ESA Diagnosis Reference*.

Chapter 20. Program manager domain

This chapter describes the program manager domain introduced in CICS/ESA 4.1. It covers the following topics:

- Overview
- Benefits of the program manager domain
- Changes to CICS externals
- Problem determination.

Overview

The program manager domain provides support for the following areas of CICS:

- Program control functions:

  EXEC CICS LINK
  EXEC CICS XCTL
  EXEC CICS LOAD
  EXEC CICS RELEASE
  EXEC CICS RETURN

- Transaction ABEND and condition handling functions:

  EXEC CICS ABEND
  EXEC CICS HANDLE ABEND
  EXEC CICS HANDLE CONDITION
  EXEC CICS HANDLE AID

- Related functions such as invoking user-replaceable modules, global user exits, and task-related user exits.
- Autoinstall for programs, mapsets, and partitionsets (see Chapter 14, “Autoinstall for programs, mapsets, and partitionsets” on page 249 for further information).

Benefits of the program manager domain

The program manager domain is part of the CICS restructure, the aim of which is to improve the reliability and availability of CICS. The main benefits are in the autoinstall facility for programs, mapsets, and partitionsets; this is described in Chapter 14, “Autoinstall for programs, mapsets, and partitionsets” on page 249.

Changes to CICS externals

There are changes to CICS externals:

- Changes to system definition
- Changes to the application programming interface
- Changes to the system programming interface
- Changes to CICS-supplied transactions.

Changes to system definition

There are two new system initialization parameters introduced for the program manager domain. These are shown in Table 29.

Table 29. The DFHSIT macro parameters

	DFHSIT	[TYPE={**CSECT**|DSECT}] ... [SPCTRPG={(1[,2][,3])|ALL|OFF}] [STNTRPG={(1],2]],3])|ALL|OFF}] ...
	END	DFHSITBA

SPCTRPG={(1[,2][,3])|ALL|OFF}
Sets the level of special tracing for the program manager domain.

See the *CICS/ESA System Definition Guide* for information about specifying special trace levels using the SPCTRxx system initialization parameters.

STNTRPG={(1[,2][,3])|ALL|OFF}
Sets the level of standard tracing for the program manager domain.

See the *CICS/ESA System Definition Guide* for information about specifying standard trace levels using the STNTRxx system initialization parameters.

Changes to the application programming interface

Two new options, INVOKINGPROG and RETURNPROG, are provided on the EXEC CICS ASSIGN command to allow an application program to determine the name of the program that linked or transferred control to it, and the name of the program to which control returns on completion.

There are also changes to:

- COMMAREA processing
- The handling of the length error (LENGERR) condition
- Transaction identifier processing.

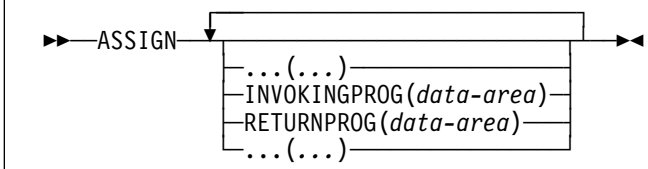
EXEC CICS ASSIGN command

General-use programming interface

Function

Request values from outside the application program's local environment.

Syntax



Conditions:
INVREQ

ASSIGN options

INVOKINGPROG(data-area)

returns the 8-character name of the application program that used the LINK or XCTL command to link or transfer control to the current program.

If you issue the ASSIGN INVOKINGPROG command in a remote program that was invoked by a distributed program link (DPL) command, CICS returns the name of the program that issued the DPL command.

If you issue the ASSIGN INVOKINGPROG command in an application program at the highest level, CICS returns eight blanks.

If you issue the ASSIGN INVOKINGPROG command in a global user exit, a task-related user exit, a user-replaceable module, or a program list table program, CICS returns eight blanks.

RETURNPROG(data-area)

returns the 8-character name of the program to which control is to be returned when the current program has finished executing. The values returned depend on how the current program was given control, as follows:

- If the current program was invoked by a LINK command, including a distributed program link, RETURNPROG returns the same name as INVOKINGPROG.
- If the current program was invoked by an XCTL command, RETURNPROG returns the name of the application program in the chain that last issued a LINK command. For example:

```
Program A links to program B

Program B links to program C

Program C transfers control to program D

Program D issues an ASSIGN RETURNPROG command,
and CICS returns the name of Program B.
```

  If the program that invoked the current program with an XCTL command is at the highest level, CICS returns eight blanks.
- If the ASSIGN RETURNPROG command is issued in the program at the top level, CICS returns eight blanks.
- If the ASSIGN RETURNPROG command is issued in a global user exit, task-related user exit, a user-replaceable module, or a program list table program, CICS returns eight blanks.

INVOKINGPROG and RETURNPROG are supported for distributed program link (DPL) in CICS/ESA 4.1, but when a pre-CICS/ESA 4.1 client links to a CICS/ESA 4.1 server by means of DPL, the CICS/ESA 4.1 server returns eight blanks as the name of the invoking program and also as the name of the program to be returned to.

Changes to COMMAREA processing

There are some changes to the rules governing the processing of COMMAREAs on EXEC CICS LINK, XCTL, and RETURN commands, with some additional error checking.

In CICS/ESA 4.1, CICS returns a LENGERR (with RESP2=26) for LINK, XCTL, and RETURN commands that specify:

- A zero address and a valid length, or
- A zero address and a negative length.

For full details of all the changes to COMMAREA processing, see the *CICS/ESA Migration Guide*.

Note: CICS also returns LENGERR if a negative length is passed for an INPUTMSG on an EXEC CICS LINK, XCTL or RETURN command.

Changes to transaction identifier processing

The next transaction identifier is cleared in the following circumstances:

- For COMMAREA errors on the final return to CICS
- INPUTMSG errors on the final return to CICS
- On abnormal termination of the transaction.

Changes to the system programming interface

There are some general changes to the system programming interface. These are:

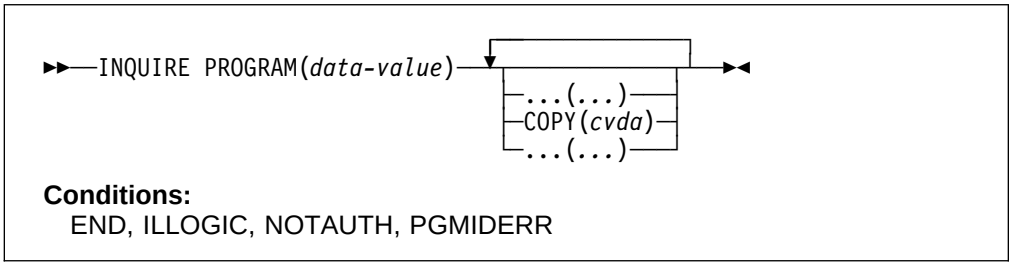
- A new option on the EXEC CICS INQUIRE PROGRAM command
- New trace component codes for the program manager domain
- Changes to EXEC CICS DISCARD command processing
- Changes to EXEC CICS SET PROGRAM command processing
- Changes to EXEC CICS INQUIRE PROGRAM LANGUAGE command processing.

EXEC CICS INQUIRE PROGRAM command

Function

To retrieve information about a program, mapset, or partitionset.

Syntax



Conditions:
END, ILLOGIC, NOTAUTH, PGMIDERR

INQUIRE PROGRAM options

COPY(cvda)

returns a CVDA value indicating the load status of the specified program. CVDA values are:

NOTREQUIRED
: The program status is either LOADABLE or NOT_LOADED, and a SET PROGRAM NEWCOPY|PHASEIN operation is not required.

REQUIRED
: The program status is NOT_LOADABLE.

  This means that a search for the program failed during a load operation, and the program has been marked as not loadable to avoid the overhead of further load attempts.

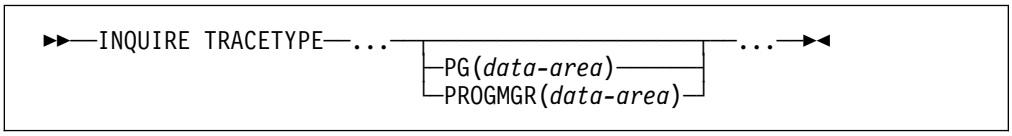
  A SET PROGRAM COPY(NEWCOPY|PHASEIN) command is required to reset the status to NOT_LOADED, which will enable the program to be loaded. However, before issuing this command, you should ensure there is a copy of the program in the DFHRPL library concatenation.

EXEC CICS INQUIRE TRACETYPE command

Function

Retrieve information on trace levels for a named CICS component.

Syntax



INQUIRE TRACETYPE options

PG(data-area) or PROGMGR(data-area)

returns the trace levels for the program manager domain, in the form of a 32-bit string, where the bit positions correspond to the trace levels.

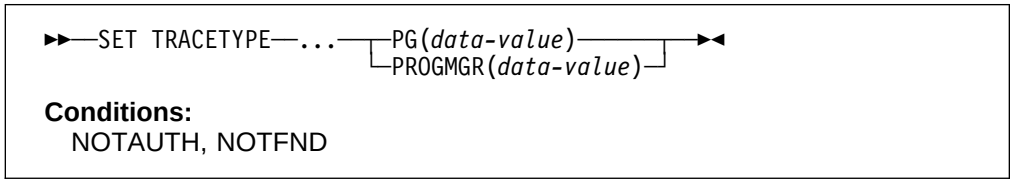
CICS returns either the standard or special flags, depending on the SPECIAL or STANDARD option specified on the command.

EXEC CICS SET TRACETYPE command

Function

Change level of tracing for a named CICS component.

Syntax



Conditions:
NOTAUTH, NOTFND

SET TRACETYPE options

PG(data-value) or PROGMGR(data-value)

sets the trace levels for the program manager domain, in the form of a 32-bit string, where the bit positions correspond to the trace levels.

CICS sets either the standard or special flags, depending on the SPECIAL or STANDARD option specified on the command.

Changes to EXEC CICS DISCARD PROGRAM processing

In earlier releases of CICS/ESA, CICS checks the table of installed transaction definitions and disallows the DISCARD PROGRAM command if the program is the initial program for a transaction.

In CICS/ESA 4.1, this check is not made, and therefore the INVREQ response with a RESP2 value of 12 is no longer returned to an application.

If you discard a program that is referenced by an installed transaction definition, an error results when the relevant transaction is run, indicating that the program can't be found. However, if autoinstall is active, the discarded program is autoinstalled again the first time a transaction needs it.

In CICS/ESA 4.1, the following RESP2 values are not returned on the INVREQ condition:

RESP2	Meaning
12	The program is named in the program control table (PCT)
13	The program is named in the program list table (PLT)
14	CICS is unable to load the PLT
16	The deletion of the program is unsuccessful

If a program deletion fails for reasons other than those listed in the *CICS/ESA System Programming Reference* manual, CICS abends the transaction with an AEXZ abend.

Addition of program load status

When a request is made for a program (for example, a LINK or a LOAD command is issued), CICS searches the DFHRPL concatenation of libraries for the requested program. To remove the need for CICS to repeat time-consuming library searches for programs which are known not to be in the library, a program load status is introduced. Program manager checks this load status before searching the library for the requested program.

The load status of a program is maintained in the program's definition as one of the following values:

Status	Meaning
LOADABLE	means that the program has been successfully installed.
NOT_LOADED	means that only the program definition has been installed.
NOT_LOADABLE	means that the search for the program has failed, and a PGMIDERR condition is returned with EIBRESP2=3. NOT_LOADABLE is reset to NOT_LOADED following a SET PROGRAM(prgmid) NEWCOPY or SET PROGRAM(prgmid) PHASEIN command. When the status is found to be NOT_LOADABLE, program manager rejects the request without calling the loader domain to load the program.

Changes to EXEC CICS SET PROGRAM processing

There are three additional reasons why CICS can raise the INVREQ condition, each with a new EIBRESP2 value, as follows:

Condition	RESP2	Meaning
INVREQ	17	You have specified an invalid option for a remote program. You cannot specify any of the following for a remote program: CEDFSTATUS (or CEDF or NOCEDF); EXECUTIONSET (or DPLSUBSET or FULLAPI); SHARESTATUS (or PRIVATE or SHARED); or COPY (or NEWCOPY or PHASEIN).
INVREQ	18	You have specified an invalid option for a mapset. You cannot specify any of the following for a mapset: CEDFSTATUS (or CEDF or NOCEDF); EXECUTIONSET (or DPLSUBSET or FULLAPI);
INVREQ	19	You have specified an invalid option for a partitionset. You cannot specify any of the following for a partitionset: CEDFSTATUS (or CEDF or NOCEDF); EXECUTIONSET (or DPLSUBSET or FULLAPI);

Changes to EXEC CICS INQUIRE PROGRAM LANGUAGE processing

APAR PN79813

Documentation for PN79813 added on 23 July 1996

On an EXEC CICS INQUIRE PROGRAM LANGUAGE command, the language is the defined language, taken from the resource definition. On an EXEC CICS

+ INQUIRE PROGRAM LANGDEDUCED command, the language is that in which the
+ module is written if known, or the defined language from the resource definition if
+ not.

+ CVDA values returned are:

+ - COBOL, LE370, C, PLI or PL1, and ASSEMBLER for supported languages
+ - NOTAPPLIC for remote programs
+ - NOTDEFINED when the program definition does not specify a language.

+ RPG is not supported, and so cannot be returned after an EXEC CICS INQUIRE
+ PROGRAM LANGUAGE command. If CICS cannot determine a language, the
+ language in the program definition is returned.

End of General-use programming interface

Changes to CICS-supplied transactions

A new component code, PG, is added to the component trace options display of the CETR transaction, as shown in Figure 21.

```
CETR                    Component Trace Options               P

Overtype where required and press ENTER.                   PAGE
Component Standard                          Special
--------  --------------------------------  ------------------

  AP      1-2                               OFF
  BF      1                                 1
  BM      1                                 1
  CP      1-2                               OFF
  DC      OFF                               OFF
  DI      1                                 1
  DM      1                                 1-2
  DS      1                                 1-2
  DU      1                                 1-2

⋮
| PG      1                                 1

PF:  1=Help   3=Quit   7=Back  8=Forward  9=Messages   ENTER=Ch
```

Figure 21. Component Trace Options screen

Problem determination

There are changes to CICS messages and abend codes, and improvements to CICS system and transaction dump formatting to aid problem determination in the program manager domain.

There are new trace entries for the program manager domain, and these are described in the *CICS/ESA Diagnosis Reference*.

Changed messages and codes

A new message, DFHAP1212, is issued when CICS detects that the language of a program is defined incorrectly.

Messages output following a CEMT DISCARD PROGRAM command are changed from DFHPC01xx messages to DFHPG01xx messages. They are still written to the CSPL transient data queue.

The following transaction abend codes are added:

ALIA A request to the storage manager to obtain storage for the OS/VS COBOL working storage and task global table areas has failed with a response other than PURGED.

ALIB A request to the storage manager to obtain storage for the C/370 run unit work area has failed with a response other than PURGED.

ALIC A request to the storage manager to obtain storage above 16MB for the LE/370 run unit work area has failed with a response other than PURGED.

ALID A request to the storage manager to obtain storage below 16MB for the LE/370 run unit work area has failed with a response other than PURGED.

ALIE A request to the storage manager to obtain storage for the C/370 thread work area has failed with a response other than PURGED.

ALIF A request to the storage manager to obtain storage for the LE/370 thread work area has failed with a response other than PURGED.

ALIG CICS is unable to determine the language of the user program.

Message tables in both English and Kanji are provided for the program manager domain.

For details of all the new, changed, and obsolete messages, see the *CICS/ESA Migration Guide*.

Changes to the formatting of system dumps

The PG keyword on the system dump causes the program manager state within the dump to be formatted.

The following control blocks are formatted in the system dump.

- The PG anchor block
- Program definitions (PPT entries)
- The system and task load list areas (previously formatted under XM component)
- Certain task-related information (for example, the program name for each logical level)
- The handle tables (for HANDLE CONDITION, HANDLE AID, HANDLE ABEND).

The installed program definitions continue to be dumped in a transaction dump when requested by EXEC CICS DUMP TRANSACTION PPT command.

Changes to the formatting of transaction dumps

When you specify the PROGRAM option on an EXEC CICS DUMP TRANSACTION DUMPCODE(name) command, the following information is provided by the program manager.

- The program level control blocks (PLCBs)
- The program transaction areas (PTAs)
- Areas related to each logical level in the task, including:
  - All COMMAREAs (formatting now indicates the program to which each commarea was passed)
  - All program entry points
  - The program level control block (PLCB).

Other details are provided by the kernel and the loader as in CICS/ESA 3.3.

Chapter 21. Transaction manager domain

This chapter describes the changes in CICS/ESA 4.1 that are associated with the introduction of the new transaction manager domain. It covers the following topics:

- Overview
- Benefits of transaction manager
- Changes to CICS externals
- Problem determination.

Overview

The transaction manager domain provides transaction-related services to:

- Create tasks
- Terminate tasks
- Purge tasks
- Inquire on tasks
- Manage transaction definitions
- Manage transaction classes.

The transaction manager domain also provides a transaction environment that enables other CICS components to implement transaction-related services.

The new transaction manager domain is designed to provide greater reliability and improved function; it has minimal impact on end users.

The restructuring of transaction manager is reflected in the reorganization of statistics. For example, much of what was reported in dispatcher statistics in earlier releases is now reported in transaction manager statistics.

New attributes on transaction resource definition

Four new options are added to the transaction definition. They are STORAGECLEAR, RUNAWAY, SHUTDOWN, and TRANCLASS.

The new TRANCLASS attribute replaces the old TCLASS attribute of transactions (although the TCLASS continues to be supported for compatibility purposes). The new transaction classes are identified by eight-character names, and are represented by TRANCLASS resource definitions, of which you can define as many as you need. New commands are provided to inquire, set, and discard transaction classes.

Dynamic discard and reinstall of transactions

You can now reinstall and discard transaction definitions while they are in use by a transaction. Any changes that you make do not affect the transactions that are currently attached.

System definition changes

There are changes to the meaning of the MXT and ICVR system initialization parameters.

The AMXT, CMXT, CMXTLIM, and MAXMIRS system initialization parameters have been removed.

Transaction restart

Transaction manager controls the coordination of transaction restart activity. The transaction retry program (DFHRTY) has been replaced by the transaction restart program (DFHREST) which has a substantially different interface. For information about migration to DFHREST see the *CICS/ESA Migration Guide*.

Global user exits

There are two enhancements to global user exits. A new transaction-attach global user exit (XXMATT) is introduced, which is invoked when a user transaction is attached. This exit is can change some of the attributes of the transaction to be attached.

The existing global user exit (XKCREQ) supports only the following functions:

- ENQUEUE
- DEQUEUE

Enhancements to simplify dynamic transaction routing

Transaction manager also eliminates the need for a specific transaction resource definition in a terminal-owning region when it is to be dynamically routed to an application-owning region.

For details of this enhancement, see “Elimination of need for transaction definitions in a terminal-owning region” on page 492.

Benefits of the transaction manager domain

Some of the benefits of transaction manager, and its associated support, are:

- You can now have an unlimited number of transaction classes. The old limit of 10 transactions classes (TCLASS) has been removed and each transaction class can be defined with an eight-character name. The current 10 transaction classes continue to be supported. Transaction classes are now represented by TRANCLASS resource definitions, and you can define as many as you need.
- There is much greater flexibility in the control of transactions and transaction classes, with new commands to inquire, set, and discard them.

  You can now reinstall and discard transaction definitions while they are in use by a transaction. Any changes that you make do not affect the transactions that are currently running.

  Also, you can now specify a runaway time limit at the transaction definition level, overriding the system default value set by a system initialization parameter.

- System definition is simplified, with the system initialization parameter MXT now applicable to user tasks only, and the removal of AMXT, CMXT, and MAXMIRS.

  Also, if you alter the MXT value to a lower limit while CICS is running, the associated kernel stack storage is freed, whereas in earlier releases of CICS/ESA the storage remains allocated as unused kernel segments.
- Greater reliability for the restart process is provided by the new transaction restart program, DFHREST.
- Enhancements to the transaction manager global user exits give you much greater flexibility and control at the point of transaction attach.
- There are new statistics for transaction classes.

Changes to CICS externals

In CICS/ESA 4.1 the changes to transaction manager result in a number of changes to CICS externals: These are:

- Changes to system definition
- Changes to resource definition
- Changes to the system programming interface (SPI)
- Changes to CICS-supplied transactions
- Changes to global user exits
- Changes to the exit programming interface (XPI)
- Changes to user-replaceable modules
- Changes to statistics.

Changes to system definition

There are additions to CICS system initialization parameters in CICS/ESA 4.1 to support the transaction manager domain, and some parameters of CICS/ESA 3.3 are now obsolete.

The system initialization parameters that are changed or made obsolete by the new transaction manager domain are shown in Table 30.

Table 30. The DFHSIT macro parameters

	DFHSIT	[TYPE={**CSECT**|DSECT}]
		.
		~~[,AMXT={**MXT-number**|number}]~~
		.
		~~[,CMXT=([n1][,n2]...)]~~
		~~[,CMXTLIM=([n1][,n2]...,n10)]~~
		.
		[,ICVR={**5000**|number}]
		[,MXT={**5**|number}]
		.
		~~.[,MAXSMIR={**999**|number}]~~
		[SPCTRXM={(1[,2][,3])|ALL|OFF}]
		[STNTRXM={(1],2]],3])|ALL|OFF}]
		.
	END	DFHSITBA

The system initialization parameters STNTRxx and SPCTRxx now support the additional code XM to specify transaction manager trace options.

The meaning of ICVR has changed. It now applies only to tasks that use the system ICVR.

The meaning of MXT has changed.

ICVR={5000|number}
specifies the default runaway task time interval in milliseconds as a decimal number. You can code zero, or a number in the range 500 through 2 700 000, in multiples of 500. CICS rounds down values that are not multiples of 500. This is the RUNAWAY interval used by transactions defined with RUNAWAY=SYSTEM (see the *CICS/ESA Resource Definition Guide*). CICS may purge a task if it has not given up control after the RUNAWAY interval for the transaction (or ICVR if the transaction definition specified RUNAWAY=SYSTEM). If you code ICVR=0, runaway task control is inoperative for transactions specifying RUNAWAY=SYSTEM in their transaction definition (that is, tasks do not get purged if they appear to be looping). The ICVR value is independent of the ICV value, and can be less than the ICV value. Note that CICS runaway task detection is based upon task time, that is, the interval is decremented only when the task has control of the processor. For information about commands that reinitialize the ICVR value, see the *CICS/ESA Problem Determination Guide*.

MXT={5|number}
Specifies the maximum number, in the range 1 through 999, of ***user*** tasks CICS allows to exist at any time. CICS queues requests for tasks above this number but does not action (attach) them until the number of tasks attached drops below the MXT limit.

Note: The MXT value does **not** include CICS system tasks.

Changes to resource definition

There are changes to the transaction resource definition and a new resource definition type, TRANCLASS, is introduced.

CICS-supplied definitions for TRANCLASS

To ease migration to the new transaction classes, CICS maps the old TCLASS attributes to new CICS-supplied TRANCLASS definitions. These are DFHTCL01 through DFLTCL10, which correspond to the TCLASS values 1 through 10.

The TRANCLASS resource definitions for the DFHTCLnn transaction classes are supplied in the CSD group, DFHTCL, which is included in DFHLIST when you initialize, or upgrade, your CSD.

When you install existing transaction definitions that specify old TCLASS numbers, CICS converts these to reference the new TRANCLASS definitions that correspond to the TCLASS number.

The TRANSACTION resource definition

There are four new attributes on the transaction resource definition:

- RUNAWAY
- SHUTDOWN
- STORAGECLEAR
- TRANCLASS

See “Changes to the TRANSACTION definition” on page 616 for details of the full DEFINE panel for the TRANSACTION resource definition.

RUNAWAY(SYSTEM|number)

The amount of time, in milliseconds, for which any task running under this transaction definition can have control of the processor before it is assumed to be in a runaway condition (logical loop). When this interval expires, CICS can abnormally terminate the task.

SYSTEM

Specifies that CICS is to use the ICVR system initialization parameter value as the runaway time limit for this transaction.

number

Specifies the runaway time limit in the range 0—2 700 000.

If you specify 0 (zero) it means there is no limit and that no runaway task detection is required for the transaction.

SHUTDOWN(DISABLED|ENABLED)

indicates whether the transaction can be run during CICS shutdown. This supplements the XLT option on EXEC CICS PERFORM SHUTDOWN, so in order for a transaction to be attached during shutdown it must either be defined as SHUTDOWN(ENABLED) or named in the XLT specified in the EXEC CICS SHUTDOWN command.

DISABLED

The transaction is disabled from running during CICS shutdown.

ENABLED

The transaction is enabled to run during CICS shutdown.

STORAGECLEAR(NO|YES)

indicates whether task-lifetime storage for this transaction is to be cleared or not upon release. This can be used to prevent other tasks accidentally viewing any confidential or sensitive data that was being stored by this transaction in task lifetime storage.

TRANCLASS(name)

is the name of the transaction class to which the transaction belongs. If a transaction class is not specified for the transaction, it is assigned to the special default transaction class, DFHTCL00. Note that this is a default name only—there is no actual CICS-supplied transaction class definition for DFHTCL00.

The TRANCLASS resource definition

See “The new TRANCLASS definition” on page 620 for details of the DEFINE panel for TRANCLASS. The attribute keywords of the TRANCLASS resource definition are described as follows:

GROUP(name)

Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and ¢. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

MAXACTIVE(value)

This is the maximum number of transactions in this TRANCLASS that are allowed to be active. You must specify a MAXACTIVE value when you define a TRANCLASS, in the range 0 through 999.

New transactions attached when the number of active transactions has reached the MAXACTIVE limit are considered for queueing subject to the PURGETHRESH limit.

Defining a TRANCLASS with a zero MAXACTIVE value signifies that **all** tasks are to be queued.

PURGETHRESH({NO|number})

This is an optional purge threshold for the transaction class; it defines a threshold number at which transactions queuing for membership of the transaction class are purged. Specify it if you want to limit the number of transactions queueing in this transaction class. It can have the following values:

NO

The size of the queue is unlimited (other than by the storage available to attach tasks).

number

The purge threshold number in the range 1—1 000 000.

If you specify this as 1, no transactions are allowed to queue. If you specify it as any other number (*n*), the size of the queue is restricted to *n–1*. All new transactions attached after the limit of *n–1* is reached are purged.

Example of PURGETHRESH: In the case of a transaction class where the maximum number of active tasks (MAXACTIVE) is set to 50, and the purge threshold (PURGETHRESH) is set to 10 to limit queuing transactions, CICS begins to abend new transactions for the class when:

- The number of active transactions reaches 50, and
- The number of transactions queuing for membership of the transaction class has reached 9.

CICS accepts new transactions for this transaction class queue only when the number queued falls below the maximum size of the queue (9 in our example).

TRANCLASS(name)
is the name of the transaction class. Transactions belonging to a transaction class are subject to scheduling constraints before they are allowed to execute. The reserved TRANCLASS name DFHTCL00 is used to indicate that the transaction does not belong to any transaction class.

For compatibility with releases that support a TCLASS attribute, CICS provides the following TRANCLASS equivalents:

TCLASS	TRANCLASS
NO	DFHTCL00
1	DFHTCL01
2	DFHTCL02
3	DFHTCL03
4	DFHTCL04
5	DFHTCL05
6	DFHTCL06
7	DFHTCL07
8	DFHTCL08
9	DFHTCL09
10	DFHTCL10

Sample definitions for these transaction classes are in group DFHTCL, supplied as part of DFHLIST.

Note: If a transaction is run and its associated TRANCLASS definition is not installed, the transaction will run without any of the scheduling constraints specified in the TRANCLASS. Message DFHXM0212 is issued as a warning.

TRANCLASS can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters.

Note: Defining a TRANCLASS with PURGETHRESH(NO) signifies that the number of waiting transactions can be unlimited. A purge threshold value of zero is shown as NO on the CEDA define TRANCLASS screen.

If the MAXACTIVE value is set to 50 and the PURGETHRESH value is set to 10 to limit the number of transactions waiting for membership of the TCLASS, the tenth waiting transaction is abended.

Changes to the system programming interface (SPI)

There are changes to the EXEC CICS INQUIRE and SET commands as part of the transaction manager domain restructure. The commands affected are:

- EXEC CICS INQUIRE TASK
- EXEC CICS SET TASK
- EXEC CICS INQUIRE TRACETYPE
- EXEC CICS SET TRACETYPE
- EXEC CICS INQUIRE TRANSACTION
- EXEC CICS SET TRANSACTION
- EXEC CICS INQUIRE TRANCLASS
- EXEC CICS SET TRANCLASS
- EXEC CICS DISCARD TRANCLASS

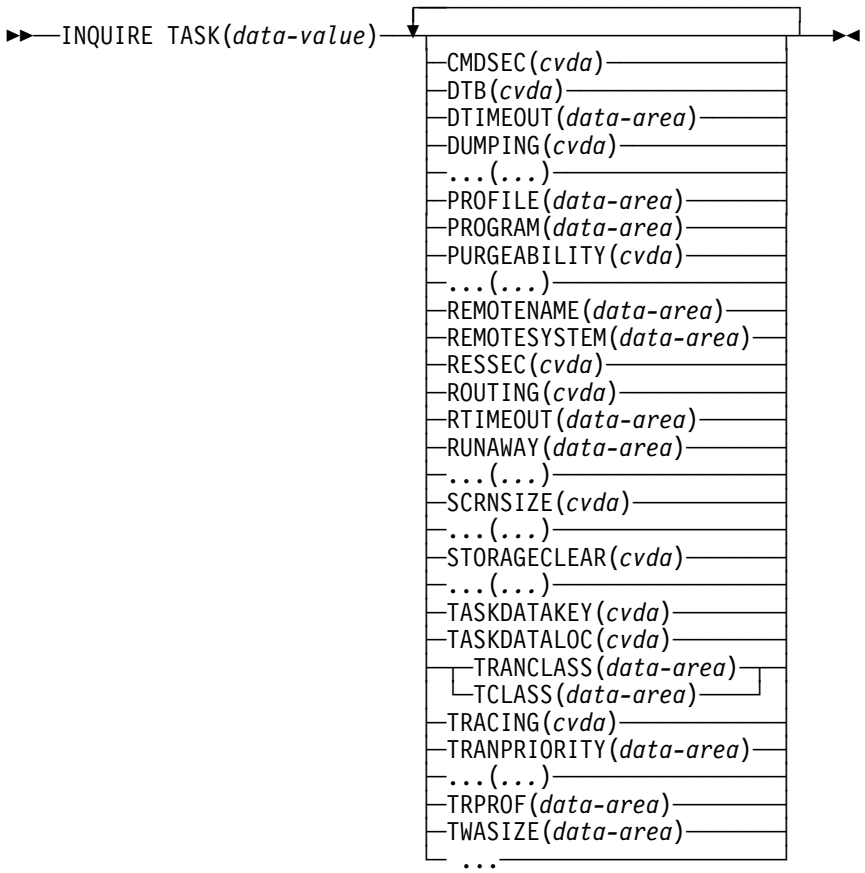
EXEC CICS INQUIRE TASK command

General-use programming interface

Function

The INQUIRE TASK command returns information about user tasks. User tasks are tasks associated with user-defined transactions and tasks associated with the CICS-supplied transactions that are normally invoked by an operator.

Syntax



Conditions:
NOTAUTH, TASKIDERR

Note that many of the options available on this command are the same as those available on the INQUIRE TRANSACTION command. However, the values returned when you inquire on a specific task can be different from those returned by an inquiry on the transaction definition associated with the task. This is because you can change a transaction definition while tasks attached using that transaction definition are still running. However, changing the attributes of an installed transaction definition does not affect those tasks that are already attached.

This means that, once attached with the attributes from its transaction definition, a task is entirely independent of that transaction definition. You can even delete a

transaction definition without having any effect on existing tasks that were associated with that definition.

INQUIRE TASK options

The following are the new options added to INQUIRE TASK by the transaction manager domain restructure.

CMDSEC(cvda)
returns a CVDA value identifying whether command security checking is in effect for this task. CVDA values are:

CMDSECNO
Command security checking is not in effect.

CMDSECYES
Command security checking is being carried out by an external security manager, such as RACF.

DTB(cvda)
returns a CVDA value identifying how uncommitted changes made to recoverable resources by this task are handled if the task fails. CVDA values are:

BACKOUT
Uncommitted changes made to recoverable resources by this task are backed out.

COMMIT
Uncommitted changes made to recoverable resources by this task are committed.

WAIT
Uncommitted changes made to recoverable resources by this task are put in a WAIT state.

DTIMEOUT(data-area)
returns a fullword binary field indicating the deadlock time-out interval (in seconds) when this task suspends.

DUMPING(cvda)
returns a CVDA value identifying whether transaction dumps are taken if the task terminates abnormally. CVDA values' are:

NOTRANDUMP
Transaction dumps are not taken.

TRANDUMP
Transaction dumps are taken.

The setting of this option is ignored when an explicit EXEC CICS DUMP TRANSACTION command is issued.

PROFILE(data-area)
returns an 8-character string indicating the name of the profile for the task.

PROGRAM(data-area)
returns an 8-character string indicating the name of the first program to be executed by this task.

PURGEABILITY(cvda)
returns a CVDA value identifying whether the task is PURGEABLE or NOTPURGEABLE in system stall conditions. CVDA values are:

NOTPURGEABLE
The task is not purgeable in system stall conditions.

PURGEABLE
The task is purgeable in system stall conditions.

REMOTENAME(data-area)
returns a 4-character string indicating the remote name of the transaction associated with this task if it is defined to run in a remote system.

For example, if the task relates to a transaction that has been routed to a remote CICS, this option provides the name of the transaction as it is known in the remote system.

REMOTESYSTEM(data-area)
returns a 4-character string indicating the name of the remote system where the transaction associated with this task is defined to run.

RESSEC(cvda)
returns a CVDA value identifying whether resource security checking is in effect for this task. CVDA values are:

RESSECNO
Resource security checking is not in effect.

RESSECYES
Resource security checking is being carried out by an external security manager, such as RACF.

ROUTING(cvda)
returns a CVDA value identifying whether the task may be or may have been subjected to dynamic routing. CVDA values are:

DYNAMIC
The task can be or could have been routed dynamically depending on whether it has been queued for initial dispatch, or attached and dispatched.

STATIC
The task is static.

RTIMEOUT(data-area)
returns, as a fullword binary field, the read time-out value, which is the number of seconds after which this task is terminated if no input is received.

RUNAWAY(data-area)
returns a fullword binary field indicating the amount of time, in milliseconds, for which any task can have control of the processor before it is assumed to be in a runaway condition (logical loop). When this interval expires, the task is abnormally terminated. The value can be 0 (zero), meaning that no runaway task detection is required.

SCRNSIZE(cvda)
returns a CVDA value identifying whether the alternate or the default screen size will be used by this task. CVDA values are:

ALTERNATE
The alternate screen size will be used by this task.

DEFAULT
The default screen size will be used by this task.

STORAGECLEAR(cvda)
returns a CVDA value identifying whether the storage for this transaction will be cleared upon release. This can be used to prevent other tasks accidentally viewing confidential data. are: CVDA values

CLEAR
The storage for this transaction will be cleared upon release.

NOCLEAR
The storage for this transaction will not be cleared upon release.

TASKDATAKEY(cvda)
returns a CVDA value identifying the storage key in which CICS obtains all storage for use by the task. This includes the task life-time storage—the transaction work area (TWA) and the EXEC interface block (EIB)—and the storage that CICS obtains on behalf of programs that run under this task.

See the description of the TASKDATAKEY parameter on the transaction resource definition in the *CICS/ESA Resource Definition Guide* for more information.

CVDA values are:

CICSDATAKEY
CICS obtains storage for the task from CICS-key storage. Application programs that execute in CICS key have read-write access to this storage, but user-key programs have read-only access.

USERDATAKEY
CICS obtains storage for the task from user-key storage. Application programs that execute in any key have read-write access to this storage.

The value of TASKDATAKEY returned by EXEC CICS INQUIRE TASK is taken from the task's transaction definition and does not take into account whether storage protection is enabled. To determine whether the task's storage is truly user key (key 9), it is necessary to determine the status of storage protection by issuing an EXEC CICS INQUIRE SYSTEM STOREPROTECT.

TASKDATALOC(cvda)
returns a CVDA value identifying whether certain CICS control blocks (including EIB and TWA) for the task are acquired above or below 16MB. CVDA values are:

ANY
The task accepts task-related data either below or above 16MB.

BELOW
The task requires any task-related data (TWA and EIB plus any internal control blocks) to be located below 16MB.

TCLASS(data-area)
returns a fullword binary field indicating whether the task belongs to a transaction class (only if numbered 1 through 10). This option is retained for compatibility purposes only. If the task does not belong to a class, the returned value is zero.

TRACING(cvda)
returns a CVDA value identifying the type of tracing for this task. CVDA values are:

SPECTRACE
Tracing for this task is special.

SPRSTRACE
Tracing for this task is suppressed.

STANTRACE
Tracing for this task is standard.

TRANCLASS(data-area)
returns an 8-character string indicating the transaction class to which the task belongs. If the transaction does not belong to a class, DFHTCL00 is returned.

TRANPRIORITY(data-area)
returns a fullword binary field indicating the priority of the transaction definition that was used to attach the task.

TRPROF(data-area)
returns an 8-character string indicating the name of the profile that is used for transaction routing.

TWASIZE(data-area)
returns a fullword binary field indicating the size of the associated transaction work area (TWA) in bytes.

Condition	*RESP2*	*Meaning*
TASKIDERR	1	The named task cannot be found.
NOTAUTH	100	The use of this command is not authorized.

EXEC CICS SET TASK command

No change is made to the syntax of the EXEC CICS SET TASK command. However, the action taken and the response returned as a result of an EXEC CICS SET TASK PURGED|FORCEPURGED can be different between CICS/ESA Version 3 and CICS/ESA 4.1.

PURGETYPE(cvda)
specifies whether CICS is to purge the task. CVDA values are:

FORCEPURGE
The task is to be purged immediately. However, in situations where the purging of a task would cause a system to abend, CICS attempts to defer the purge until the task reaches a state where purging the task does not harm the system.

Note: FORCEPURGE may still cause the system to terminate.

PURGE

The task can be terminated only if system and data integrity can be maintained. A task is not purged if its associated transaction definition specifies SPURGE=NO. In situations where the purging of a task would cause a system to abend, CICS defers the purge until the task reaches a state where purging the task does not harm the system.

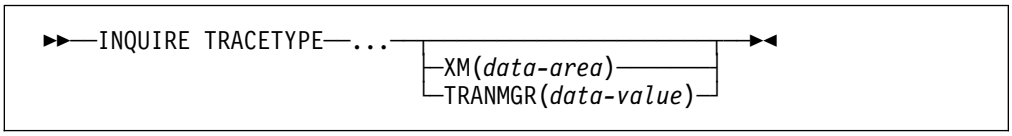
SET TASK conditions

Condition	*RESP2*	*Meaning*
NORMAL	0	Purge request has been accepted and the target task is purged.
NORMAL	13	Purge request has been accepted, but the purge is deferred.
INVREQ	6	The task is invoking DL/I and cannot be purged.

EXEC CICS INQUIRE and SET TRACETYPE command

The XM and TRANMGR options are added to the INQUIRE and SET TRACETYPE commands to allow you to inquire on, and set, trace levels for the transaction manager domain.

Syntax



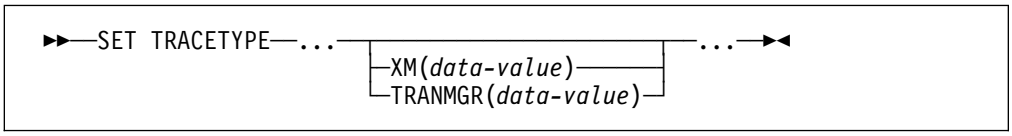
INQUIRE TRACETYPE options

XS(data-area)

returns the trace levels for the transaction manager domain in the form of a 32-bit string, where the bit positions correspond to the trace levels.

You can also specify TRANMGR as the component identifier for the transaction manager domain.

Syntax



SET TRACETYPE options

XS(data-value)

specifies the trace levels for the transaction manager domain in the form of a 32-bit string, where the bit positions correspond to the trace levels.

You can also specify TRANMGR as the component identifier for the transaction manager domain.

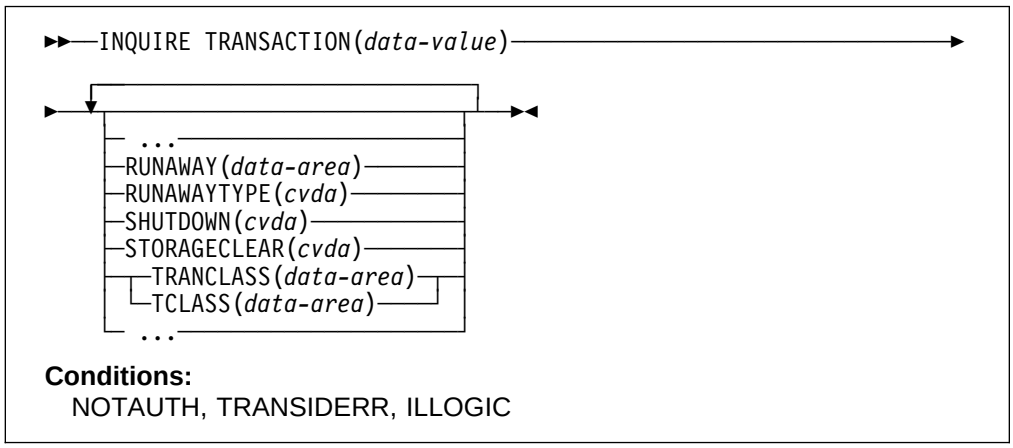
EXEC CICS INQUIRE TRANSACTION command

Several new keywords are added to the EXEC CICS INQUIRE TRANSACTION command.

Function

Retrieves information about a named transaction definition.

Syntax



Conditions:
NOTAUTH, TRANSIDERR, ILLOGIC

INQUIRE TRANSACTION options

The following are the new options added to INQUIRE TRANSACTION by the transaction manager domain restructure.

RUNAWAY(data-area)
returns a fullword binary field indicating the amount of time, in milliseconds, for which any task running with this transaction definition can have control of the processor before it is assumed to be in a runaway condition (logical loop). When this interval expires, the task is abnormally terminated. The value can be 0 (zero), meaning that no runaway task detection is required.

The units are the same as those for the ICVR system initialization parameter.

RUNAWAYTYPE(cvda)
returns a CVDA value indicating whether the transaction definition uses the current system runaway limit or one specified for the transaction. CVDA values are:

SYSTEM
The transaction uses the current system runaway limit.

USER
The transaction uses a user-defined limit specified for the transaction on the transaction definition.

SYSTEM is the default and means that the transaction uses the system default (ICVR) value.

SHUTDOWN(cvda)
returns a CVDA value identifying whether terminal-related transactions defined as SHUTENABLED can be attached during CICS shutdown. This supplements the XLT option on EXEC CICS PERFORM SHUTDOWN. CVDA values are:

SHUTDISABLED
The transaction is not allowed to be attached during shutdown.

SHUTENABLED
The transaction can be attached during shutdown.

STORAGECLEAR(cvda)
returns a CVDA value identifying whether the task-lifetime storage for a task associated with this transaction definition is cleared or not upon release. This can be used to prevent other tasks accidentally viewing confidential data. CVDA values are:

CLEAR
Storage will be cleared.

NOCLEAR
Storage will not be cleared.

TCLASS(data-area)
returns a fullword binary field indicating the number of the transaction class to which the transaction belongs (numbered 1 through 10).

This option is superseded by the 8-character TRANCLASS names for transaction classes, but TCLASS is still supported for compatibility purposes. This means that transaction resource definitions on the CSD can contain both a TCLASS number in the range 1–10 for use on earlier releases, and an 8-character TRANCLASS name.

If the transaction definition has both a TCLASS value and one of the CICS-supplied TRANCLASS names, the value returned on the INQUIRE TCLASS option is always the number that corresponds to the TRANCLASS name, regardless of the value on the TCLASS. For example, an installed transaction that is defined with TCLASS(05) and TRANCLASS(DFHTCL09) returns a TCLASS value of 09, with response NORMAL.

If the two parameters are inconsistent, and the TRANCLASS name is **not** one of the reserved names (DFHTCL01–DFHTCL10), CICS raises an exception condition (INVREQ).

TRANCLASS(data-area)
returns an 8-character string indicating the transaction class to which the transaction belongs. If the transaction does not belong to a class, DFHTCL00 is returned.

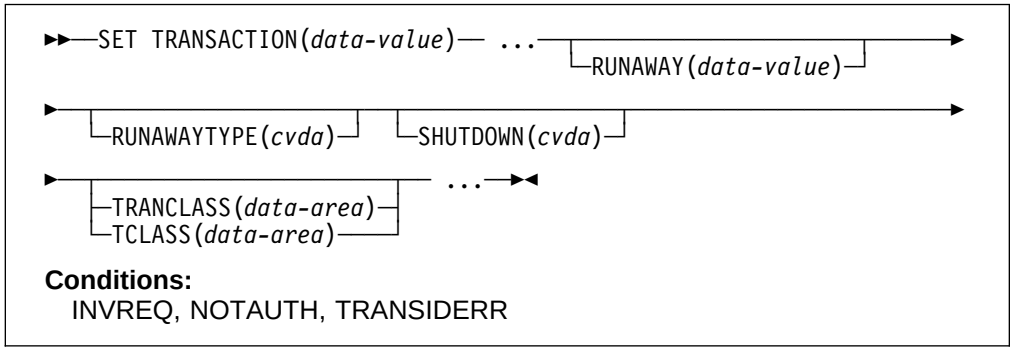
INQUIRE TRANSACTION conditions

Condition	***RESP2***	***Meaning***
ILLOGIC	1	A START has been given when a browse is already in progress, or a NEXT has been given without a preceding START.
INVREQ	3	The TCLASS operand is specified, and: • The transaction is defined with an inconsistent TCLASS number **and** TRANCLASS name, and • The TRANCLASS name is not one of the ten CICS-supplied default names, DFHTCL01 through DFHTCL10.
NOTAUTH	100	The use of this command is not authorized.
NOTAUTH	101	The access that you have requested to this transaction is not authorized *(not applicable to browse)*.
TRANSIDERR	1	The named transaction could not be found.

EXEC CICS SET TRANSACTION command

You can change the new transaction attributes with the SET command.

Syntax



RUNAWAY(data-value)
specifies, as a fullword binary variable, the amount of time, in milliseconds, for which any task running with this transaction definition can have control of the processor before it is assumed to be in a runaway condition (logical loop). When this interval expires, the task is abnormally terminated. The value can be 0 (zero), meaning that no runaway task detection is required.

The units are the same as those for the ICVR system initialization parameter.

RUNAWAYTYPE(cvda)
specifies a CVDA value identifying whether the transaction definition uses the current system runaway limit or one set by the user. CVDA values are:

SYSTEM
The current system runaway limit will be used.

USER
The limit set by the user will be used.

SYSTEM is the default and means that the transaction uses the system default (ICVR) value. To set the non-system default, the option USER should be specified for RUNAWAYTYPE, and the value itself should be set in the RUNAWAY option.

- Error cases include:
  - RUNAWAYTYPE(USER) but no RUNAWAY specified.
  - RUNAWAYTYPE(SYSTEM) but RUNAWAY specified.
  - RUNAWAY specified without RUNAWAYTYPE.

SHUTDOWN(cvda)
specifies a CVDA value identifying whether terminal-related transactions defined as SHUTENABLED can be attached during CICS shutdown. This supplements the XLT option on EXEC CICS PERFORM SHUTDOWN. CVDA values are:

SHUTDISABLED
The transaction is not allowed to be attached during shutdown.

SHUTENABLED
The transaction can be attached during shutdown.

TCLASS(data-value)

specifies, as a fullword binary variable, the number of the transaction class to which the transaction belongs (numbered 1 through 10).

This option is superseded by the 8-character TRANCLASS names for transaction classes, but TCLASS is still supported for compatibility purposes. This means that transaction resource definitions on the CSD can contain both a TCLASS number in the range 1–10 for use on earlier releases, and an 8-character TRANCLASS name.

If the transaction definition has both a TCLASS value and one of the CICS-supplied TRANCLASS names, the value returned on the INQUIRE TCLASS option is always the number that corresponds to the TRANCLASS name, regardless of the value on the TCLASS. For example, an installed transaction that is defined with TCLASS(05) and TRANCLASS(DFHTCL09) returns a TCLASS value of 09, with response NORMAL.

If the two parameters are inconsistent, and the TRANCLASS name is **not** one of the reserved names (DFHTCL01–DFHTCL10), CICS raises an exception condition (INVREQ).

TRANCLASS(data-value)

specifies, as an 8-character string, the transaction class to which the transaction belongs. If the transaction does not belong to a class, DFHTCL00 is returned.

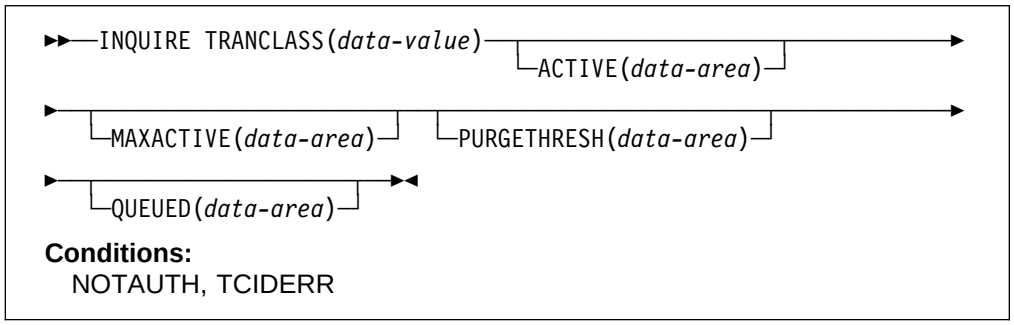
SET TRANSACTION conditions

Condition	*RESP2*	*Meaning*
INVREQ	2	PURGEABILITY has an invalid CVDA value.
INVREQ	3	STATUS has an invalid CVDA value.
INVREQ	4	Attempt to disable a CICS-supplied transaction.
INVREQ	5	TCLASS or TRANCLASS name is not known.
INVREQ	6	TCLASS or TRANCLASS was specified for a CICS-supplied transaction.
INVREQ	7	TRACING has an invalid CVDA value.
INVREQ	8	DUMPING has an invalid CVDA value.
INVREQ	9	PRIORITY value is not in the range 1–255.
INVREQ	10	RUNAWAYTYPE has an invalid CVDA value.
INVREQ	11	SHUTDOWN has an invalid CVDA value.
INVREQ	12	SET RUNAWAYTYPE(USER) has been specified but no value for RUNAWAY has been specified.
INVREQ	13	SET RUNAWAY has been used but the RUNAWAYTYPE is not USER.
INVREQ	14	SET RUNAWAY value is greater than 2 700 000 or less than –1 (–1 means ignore).
NOTAUTH	100	The use of this command is not authorized.
NOTAUTH	101	The access requested to this transaction is not authorized.
TRANSIDERR	1	The named transaction cannot be found.

EXEC CICS INQUIRE TRANCLASS command

EXEC INQUIRE TRANCLASS is the preferred option to report on the state of transaction classes, although INQUIRE TCLASS continues to be supported.

Command syntax



Conditions:
NOTAUTH, TCIDERR

The following command allows you to browse through the transaction class definitions.

Browse TRANCLASS

```
INQUIRE TRANCLASS START [AT (8-character data-value)

INQUIRE TRANCLASS(8-character data-value) NEXT
The browse options are the same as for INQUIRE TRANCLASS.

INQUIRE TRANCLASS END
```

INQUIRE TRANCLASS options

ACTIVE(data-area)
returns a fullword binary field indicating the total number of current tasks that are active in the transaction class.

MAXACTIVE(data-area)
returns a fullword binary field indicating the largest number of transactions in the transaction class that are allowed to be attached to run concurrently. The value can be in the range 0-999.

PURGETHRESH(data-area)
returns a fullword binary field indicating the limit at which queuing transactions are purged for the transaction class.

QUEUED(data-area)
returns a fullword binary field indicating the total number of current tasks that are suspended because the class maximum has been reached.

TRANCLASS(data-value)
specifies the 8-character transaction class name.

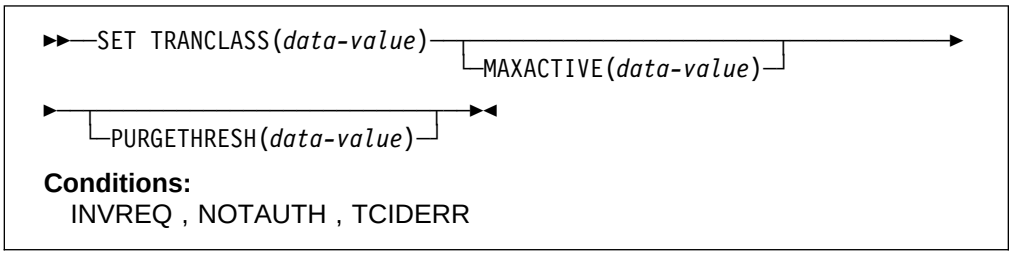
INQUIRE TRANCLASS conditions

Condition	*RESP2*	*Meaning*
NOTAUTH	101	The use of this command is not authorized.
TCIDERR	1	The named tranclass cannot be found.

EXEC CICS SET TRANCLASS command

The SET TRANCLASS command allows you to change the limits for a specified transaction class.

SET TRANCLASS



Conditions:
INVREQ , NOTAUTH , TCIDERR

SET TRANCLASS options

MAXACTIVE(data-value)

specifies a fullword binary field indicating the largest number of transactions in the transaction class that are allowed to be attached to run concurrently. The value can be in the range 0-999.

PURGETHRESH(data-value)

specifies a fullword binary field indicating the limit at which queuing transactions are purged for the transaction class.

A value of zero indicates that there is no purge threshold limit for the specified transaction class.

Effects of the SET TRANCLASS command

The SET TRANCLASS command changes the MAXACTIVE limit and/or the PURGETHRESH limit of the named transaction class.

Changing the MAXACTIVE limit

Raising the MAXACTIVE limit allows more transactions to become active in the class. If transactions are currently queueing in the class, a number of these are released from the queue and become active in the class.

Lowering the MAXACTIVE limit, however, does not have such an immediate effect. If the number of active transactions in the class is larger than the new limit, this number only decreases as the active transactions end and no new transactions are released to take their place.

Note: The MAXACTIVE limit of the transaction class can be temporarily lowered to zero, which effectively means that no transactions in that class become active during that time period.

Changing the PURGETHRESH limit

Raising the PURGETHRESH limit allows more transactions to queue waiting to become active in the class. However, the size of the queue can only increase when new transactions are attached in the class.

Lowering the PURGETHRESH limit, however, can have an immediate effect if the limit is lowered below the current size of the queue. In this case, transactions with the lowest priority in the queue are abended (AKCC) so that the new purge threshold holds for the class. A PURGETHRESH value of zero disables the purge threshold of the class and effectively means that the size of the queue is unlimited.

Note: The PURGETHRESH limit of the transaction class can be set to one, which effectively means that no transactions in that class are queued. If the MAXACTIVE limit is set to zero at the same time, all new transactions attached in that class are abended.

Changing both limits

It is possible to change both the MAXACTIVE and PURGETHRESH limits in a single command. In this instance, the change to the MAXACTIVE limit is actioned first so that preference is given to making queuing transactions active rather than abending them.

Summary of operation

The following table summarizes the effects of changing transaction class limits.

Table 31. TRANCLASS limit changes

	Raise by n	**Lower by n**
MAXACTIVE	A maximum of n new active transactions in the class	No immediate effect
PURGETHRESH	No immediate effect	A maximum of n queuing members now abended

Note: Lowering the MAXACTIVE limit and raising the PURGETHRESH have gradual, rather than immediate, effects.

SET TRANCLASS conditions

Condition	***RESP2***	***Meaning***
INVREQ	2	The MAXACTIVE value is not in the range 0–999.
INVREQ	3	The PURGETHRESH value is not in the range 0–1 000 000.
NOTAUTH	101	You are not authorized to use this command.
TCIDERR	1	The named transaction class cannot be found.

EXEC CICS DISCARD TRANCLASS command

The DISCARD TRANCLASS command removes the installed transaction class from storage and from the CICS global catalog.

You are **not** prevented from using the TCLASS option to discard the ten transaction classes provided for compatibility with previous releases. However, if you do, you can get a TCIDERR when using the INQUIRE AND SET TCLASS commands.

Command syntax

```
►►──DISCARD TRANCLASS(data-value)──►◄
```

Conditions:
INVREQ, NOTAUTH, TCIDERR

DISCARD TRANCLASS option

TRANCLASS(data-value)

specifies the name of the transaction class that you want removed from your CICS region (from the internal table and from the global catalog). The name can be up to 8-characters

DISCARD TRANCLASS conditions

Condition	*RESP2*	*Meaning*
INVREQ	12	The transaction is currently active or currently queued in a transaction class so it cannot be discarded.
NOTAUTH	100	You are not authorized to use this command.
TCIDERR	1	The named tranclass cannot be found.

EXEC CICS DISCARD TRANSACTION command

The following RESP2 values are obsolete:

12 The named transaction is currently in use

16 The deletion was unsuccessful

EXEC CICS INQUIRE and SET TRACETYPE commands

A new trace component code, XM, is added as the transaction manager component code to the EXEC CICS INQUIRE TRACETYPE and the EXEC CICS SET TRACETYPE commands.

EXEC CICS INQUIRE and SET TCLASS commands

The EXEC CICS INQUIRE TCLASS and EXEC CICS SET TCLASS commands are superseded by the new INQUIRE and SET TRANCLASS commands, as a consequence of the change from the old TCLASS to the new transaction classes (TRANCLASS). However, the old form of the commands continue to be fully supported, in this release, at both the source and object level.

You are recommended to use INQUIRE TRANCLASS and SET TRANCLASS for all new application programs, and modify any old applications when you can. The CICS translator supports both the old TCLASS and the new TRANCLASS options.

For compatibility purposes, the old commands are also supported at run time if issued in any existing application programs. CICS modifies these at execution time to map the TCLASS number to the new CICS-supplied transaction classes, DFHTCL01 through DFHTCL10. Thus if an existing application issues an EXEC CICS TCLASS(9) command, CICS returns values taken from DFHTCL09.

See the *CICS/ESA Migration Guide* for more information about migrating your old transaction classes.

End of General-use programming interface

Changes to CICS-supplied transactions

The SPI changes made to the transaction class interfaces are mirrored in the corresponding CEMT functions. Note that the existing interfaces now operate using the TCLASS keyword. TCLASS is assumed to mean TRANCLASS.

CEMT INQUIRE TRANSACTION

This CEMT screen interface changes to support named transaction classes.

```
INQUIRE TRANSACTION
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Trans(CATA) Pri( 255 ) Pro(DFHZATA ) Tcl(DFHTCL00) Ena Pur
 Trans(CATD) Pri( 255 ) Pro(DFHZATD ) Tcl(DFHTCL00) Ena Pur
 Trans(CATR) Pri( 255 ) Pro(DFHZATR ) Tcl(DFHTCL00) Ena
 Trans(CATS) Pri( 255 ) Pro(DFHZATS ) Tcl(DFHTCL00) Ena
 Trans(CBRC) Pri( 001 ) Pro(DFHBRCP ) Tcl(DFHTCL00) Ena
 Trans(TRAN) Pri( 001 ) Pro(OLDPROGA) Tcl(DFHTCL01) Ena
 Trans(ACCT) Pri( 001 ) Pro(ACCT00  ) Tcl(SESSAOR ) Ena
```

Figure 22. CEMT INQUIRE TRANSACTION panel.

Tclass(value)
displays an 8-character string identifying the transaction class to which the transaction belongs. If the transaction does not belong to a class, DFHTCL00 is returned. To remove a transaction from its transaction class, set this field to DFHTCL00. An added or changed transaction class **must** be one that has already been defined.

CEMT SET TRANSACTION

Tclass(classname)
specifies the 8-character name of the transaction class to which the transaction belongs. To remove a transaction from its transaction class, set this field to DFHTCL00. An added or changed transaction class **must** be one that has already been defined.

CEMT INQUIRE TASK

TClass(classname)
displays the 8-character name of the transaction class to which the tasks belong.

If you want to display all tasks not belonging to a transaction class, use the reserved name DFHTCL00.

CEMT INQUIRE TRANCLASS

This is a new command for inquiring about installed transaction classes.

Function

INQUIRE TRANCLASS returns information about the current and maximum number of active tasks, the purge threshold, the number of active tasks, and the number of queuing tasks of a transaction class.

Command syntax

INQUIRE TRANCLASS

```
CEMT INQUIRE TRANCLASS(classname)
[ALL]
[ACTIVE(value)]
[QUEUED(value)]
[MAXACTIVE(value)]
[PURGETHRESH(value)]
```

```
INQUIRE TRANCLASS
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Tranc(DFHTCL01) Max( 001 ) Act(000) Pur( 0000000 )
    Que(000000)
 Tranc(DFHTCL02) Max( 001 ) Act(000) Pur( 0000000 )
    Que(000000)
 Tranc(DFHTCL03) Max( 001 ) Act(000) Pur( 0000000 )
    Que(000000)
 Tranc(DFHTCL04) Max( 001 ) Act(000) Pur( 0000000 )
    Que(000000)
 Tranc(DFHTCL05) Max( 001 ) Act(000) Pur( 0000000 )
    Que(000000)
 Tranc(DFHTCL06) Max( 001 ) Act(000) Pur( 0000000 )
    Que(000000)
 Tranc(DFHTCL07) Max( 001 ) Act(000) Pur( 0000000 )
    Que(000000)
 Tranc(DFHTCL08) Max( 001 ) Act(000) Pur( 0000000 )
    Que(000000)
 Tranc(DFHTCL09) Max( 001 ) Act(000) Pur( 0000000 )
    Que(000000)
 Tranc(DFHTCL10) Max( 001 ) Act(000) Pur( 0000000 )
    Que(000000)
```

Figure 23. An example of a CEMT INQUIRE TRANCLASS panel.

Command options

ALL

For INQUIRE, this is the default.

ACTIVE(value)

displays the total number of current tasks that are active in the transaction class.

Note: All those active tasks in the transaction class are not necessarily running because they may be queued for MXT reasons.

MAXACTIVE(value)

displays the largest number of transactions in the transaction class that are allowed to be attached to run concurrently. The value can be in the range 0-999.

PURGETHRESH(value)

displays the limit at which queuing transactions are purged for the transaction class.

It can have a value in the range 0–1 000 000. Zero means that the transactions are not purged. One means that no transactions are queued. Two means that the first transaction to arrive is queued and the second is purged. Three means that the first two transactions to arrive are queued and the third is purged.

QUEUED(value)

displays the total number of current tasks that are suspended because the class maximum has been reached.

TRANCLASS(classname)

specifies the 8-character name of the transaction class.

CEMT SET TRANCLASS

SET TRANCLASS

```
CEMT SET TRANCLASS(classname)
[ALL]
[MAXACTIVE(value)]
[PURGETHRESH(value)]
```

Function

SET TRANCLASS allows you to set the maximum number of active tasks and the purge threshold of the transaction class.

Command options

ALL

Any changes you request are made to all transaction classes.

ACTIVE(value)

specifies the total number of current tasks that are active in the transaction class.

MAXACTIVE(value)

specifies the largest number of transactions in the transaction class that are allowed to be attached to run concurrently. The value can be in the range 0-999.

PURGETHRESH(value)

specifies the limit at which queuing transactions are purged for the transaction class.

It can have a value in the range 0–1 000 000. Zero means that the transactions are not purged. One means that no transactions are queued. Two means that the first transaction to arrive is queued and the second is purged. Three means that the first two transactions to arrive are queued and the third is purged.

TRANCLASS(classname)

specifies the 8-character name of the transaction class.

CEMT DISCARD TCLASS

This interface is similar to other DISCARD interfaces supported by CEMT.

DISCARD

```
CEMT DISCARD {TCLASS(tclass_name)
  ⋮
TCLASS(tclass_name)|
```

Command options

TClass(value)
specifies the 8-character name of the transaction class to be discarded.

CETR—trace control transaction

The “Component Trace Options” panel displayed by the CETR transaction is enhanced to include the XM component. Figure 24 illustrates part of a CETR component display panel, showing the new transaction manager component, XM.

```
CETR                    Component Trace Options                      PR OD1

Overtype where required and press ENTER.                          PAGE 1 OF 3
Component Standard                          Special
--------  --------------------------------  --------------------------------

 ⋮
  XM      1                                 1-2


PF:  1=Help   3=Quit   7=Back  8=Forward  9=Messages   ENTER=Change
```

Figure 24. CETR transaction: component trace options screen

Changes to global user exits

Product-sensitive programming interface

There are some changes to the global user exit interface in CICS/ESA 4.1 for the transaction manager domain.

XXMATT This is a new exit introduced in CICS/ESA 4.1.

XKCREQ The XKCREQ exit is now named the task control program exit, XKCREQ, whereas it was previously named the transaction manager program exit.

Also, XKCREQ now supports only the following functions:

- ENQUEUE
- DEQUEUE

There are new exit-specific parameters for the XKCREQ exit point.

Transaction manager program XXMATT

Exit XXMATT

		Invoked **during** transaction attach. This exit is able to change some of the attributes of the transaction that is being attached.
Exit-specific parameters	**UEPTRANID**	The address of transaction id (see Notes).
	UEPUSER	The address of the userid associated with the transaction if the current task is a user task (see Notes).
	UEPTERM	The address of the terminal id associated with the transaction, if any (see Notes).
	UEPPROG	The address of the application program name for this transaction, if any (see Notes).
	UEPATPTI	The address of a 4-byte field containing the primary transaction id. You can change the primary transaction id by modifying the addressed field.
	UEPATOTI	The address of the 4-byte attach transaction id. A transid of X'00000000' indicates that a transid was not supplied on the attach.
	UEPATTPL	The address of an area containing the length of the attach TPName. A length of zero indicates that a TPName was not supplied on the attach.
	UEPATTPA	The address of a fullword containing the address of the attach TPName. The attach TPName can be 1 through 64 bytes long, as defined by UEPTTPL.
	UEPATLOC	The address of a 1-byte field indicating whether the transaction was found. Equated values are: **UEATFND** The transaction was found **UEATNFND** The transaction was not found.
	UEPATTST	The address of a one-byte transaction definition state. Equated values for the definition state are: **UEATENAB** Transaction is enabled **UEATDISA** Transaction is disabled.
	UEPATTTK	The address of a doubleword containing a transaction token. Note that some of the transaction manager XPI calls require this token to identify the transaction that is being attached.
Return codes	**UERCNORM**	Continue attach processing.
XPI calls	The user exit can inquire on the transaction being attached using the UEPATTTK transaction token as input to the XMIQ INQUIRE_TRANSACTION XPI call. The exit can also set the total priority and tclass using the XMIQ SET_TRANSACTION XPI call. Most of the XPI calls can be used, but with caution since typically this exit is invoked under the TCP task. Thus it is advisable not to issue any XPI calls that might cause the TCP task to wait.	

Notes:

1. The following XPI calls can be useful for obtaining information that could be used to modify the attach of a transaction:

   INQUIRE_TRANSACTION
   INQUIRE_MXT
   INQUIRE_TCLASS
   INQUIRE_TRANDEF
   INQUIRE_SYSTEM

2. The fields UEPTRANID, UEPUSER, UEPTERM, and UEPPROG are common to many of the domain global user exit points, and normally return values associated with the current user task. In the case of XXMATT, however, the user task that is being attached is **not** the current task when the exit is invoked. Until task attach is complete, the current task is the CICS task that is performing the attach.

   When the task being attached is for a task started by an immediate START command; that is, a START without an interval, the current task is the task that issues the START command, and the fields contain values associated with that task.

Exit XKCREQ

	Invoked before ENQUEUE and DEQUEUE requests.	
Exit-specific parameters	**UEPENQFN**	A 1-byte field indicating the type of function—either ENQUEUE or DEQUEUE—for which this global user exit is invoked. The equated values for these functions are: **UEPENQ** The function that caused this invocation is ENQUEUE **UEPDEQ** The function that caused this invocation is DEQUEUE
	UEPENQA	The address of the resource name that is the object of the ENQUEUE or DEQUEUE request.
	UEPENQL	The length of the resource name.
Return codes	**UERCNORM**	Continue processing.
	UERCPURG	Task purged during XPI call.
XPI calls	All can be used.	

Changes to the exit programming interface

The transaction manager domain provides a number of function calls to the exit programming interface. These functions, and the macro calls that support them, are as follows:

Function	Transaction manager macro call
INQUIRE_DTRTRAN	DFHXMSRX
INQUIRE_MXT	DFHXMSRX
INQUIRE_TCLASS	DFHXMCLX
INQUIRE_TRANDEF	DFHXMXDX
INQUIRE_TRANSACTION	DFHXMIQX
SET_TRANSACTION	DFHXMIQX

There are also two new kernel domain function calls. These functions, and the macro calls that support them, are as follows:

Function	Kernel domain macro call
START_PURGE_PROTECTION	DFHKEDSX
STOP_PURGE_PROTECTION	DFHKEDSX

The transaction manager XPI calls are described in the following sections.

The INQUIRE_DTRTRAN function

The INQUIRE_DTRTRAN function is provided on the DFHXMSRX macro call. Its purpose is to provide the name of the dynamic transaction routing (DTR) transaction definition.

The DTR transaction definition provides common DTR attributes for transactions that are to be dynamically routed and which don't have a specific transaction definition. It is specified on the DTRTRAN system initialization parameter, where the CICS-supplied default name is CRTX.

The syntax of the call is shown as follows:

INQUIRE_DTRTRAN

```
DFHXMSRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_DTRTRAN),]
       [OUT,
       DTRTRAN(name4),
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

DTRTRAN(name4)
is the name of the DTR transaction definition to used for routing transactions that are not defined by an explicit transaction resource definition.

name4 The name of a 4-byte location that is to receive the name of the DTR transaction definition.

RESPONSE and REASON values for INQUIRE_DTRTRAN:

RESPONSE	*REASON*
OK	None
DISASTER	LOGIC_ERROR
	ABEND
	LOOP
INVALID	INVALID_FUNCTION
KERNERROR	None
PURGED	None

The INQUIRE_MXT function

The INQUIRE_MXT function is provided on the DFHXMSRX macro call. Its purpose is to provide current value of the MXT parameter.

The syntax of the call is shown as follows:

INQUIRE_MXT

```
DFHXMSRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_MXT),]
       [OUT,
       MXT_LIMIT(name4 | (Rn)),
       CURRENT_ACTIVE(name4 | (Rn) ),
       MXT_QUEUED(name4 | (Rn) ),
       TCLASS_QUEUED(name4 | (Rn) ),
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

MXT_LIMIT(name4 | (Rn))
is the current number of the MXT parameter.

name4 The name of a 4-byte location that is to receive the maximum number of all user tasks currently allowed, expressed as a binary value.

(Rn) A register to receive the maximum number of all tasks currently allowed, expressed as a binary value.

CURRENT_ACTIVE(name4 | (Rn))
is the current number of all active user tasks.

name4 The name of a 4-byte location that is to receive the current number of active user tasks, expressed as a binary value.

(Rn) A register to receive the current number of active user tasks, expressed as a binary value.

MXT_QUEUED(name4 | (Rn))
is the current number of user transactions that are queued as a result of the maximum tasks (MXT) being reached.

name4 The name of a 4-byte location that is to receive the current number of queued user tasks, expressed as a binary value.

(Rn) A register to receive the current number of queued user tasks, expressed as a binary value.

TCLASS_QUEUED(name4 | (Rn))
is the current number of all transactions that are queued for transaction class membership.

name4 The name of a 4-byte location that is to receive the current number of queued transaction class members, expressed as a binary value.

(Rn) A register to receive the current number of queued transaction class members, expressed as a binary value.

RESPONSE and REASON values for INQUIRE_MXT:

RESPONSE	*REASON*
OK	None
DISASTER	LOGIC_ERROR
	ABEND
	LOOP
INVALID	INVALID_FUNCTION
KERNERROR	None
PURGED	None

The INQUIRE_TCLASS function

The INQUIRE_TCLASS function is provided on the DFHXMCLX macro call. Its purpose is to provide current information about the specified transaction class (TCLASS).

The syntax of the call is shown as follows:

INQUIRE_TCLASS

```
DFHXMCLX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_TCLASS),
       INQ_TCLASS_NAME(name8 | string
       | 'string'),]
       [OUT,
       [MAX_ACTIVE(name4 | (Rn)),]
       [PURGE_THRESHOLD(name4 | (Rn)),]
       [CURRENT_ACTIVE(name4 | (Rn)),]
       [CURRENT_QUEUED(name4 | (Rn)),]]
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

INQ_TCLASS_NAME(name8 | string | 'string')
specifies the name of the transaction class for this inquiry.

name8 The name of an 8-byte location that contains the name of the transaction class being inquired on.

string A string of characters, without intervening blanks, naming the transaction class.

'string' A string of characters, within quotation marks, naming the transaction class. The string length is set to 8 by padding with blanks within the quotation marks.

MAX_ACTIVE(name4 | (Rn))
is the current maximum number of active tasks allowed for the transaction class.

name4 The name of a 4-byte location that is to receive the current maximum number of active tasks currently allowed for this transaction class, expressed as a binary value.

(Rn) A register to receive the current maximum number of active tasks currently allowed for this transaction class, expressed as a binary value.

PURGE_THRESHOLD(name4 | (Rn))
is the purge threshold limit for this transaction class.

name4 The name of a 4-byte location that is to receive the current purge threshold limit for this transaction class, expressed as a binary value.

(Rn) A register to receive the current purge threshold limit for this transaction class, expressed as a binary value.

CURRENT_ACTIVE(name4 | (Rn))
is the current number of active user tasks in this transaction class.

name4 The name of a 4-byte location that is to receive the current number of active user tasks for this transaction class, expressed as a binary value.

(Rn) A register to receive the current number of active user tasks for this transaction class, expressed as a binary value.

CURRENT_QUEUED(name4 | (Rn))
is the current number of queued user tasks.

name4 The name of a 4-byte location that is to receive the current number of queued user tasks in this transaction class, expressed as a binary value.

(Rn) A register to receive the current number of queued user tasks, expressed as a binary value.

RESPONSE and REASON values for INQUIRE_TCLASS:

RESPONSE	*REASON*
OK	None
DISASTER	LOGIC_ERROR
INVALID	None
EXCEPTION	UNKNOWN_CLASS

The INQUIRE_TRANDEF function

The INQUIRE_TRANDEF function is provided on the DFHXMXDX macro call. Its purpose is to allow you to obtain information about the specified transaction definition. In general, this function call is equivalent to the EXEC CICS INQUIRE TRANSACTION command, with some differences.

The syntax of the call is shown as follows:

INQUIRE_TRANDEF

```
DFHXMXDX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_TRANDEF),
       INQ_TRANSACTION_ID(name4 | string
       | 'string'),]
       [OUT,
       [TRANSACTION_ID(name4),]
       [INITIAL_PROGRAM(name8),]
       [PROFILE_NAME(name8),]
       [TRAN_PRIORITY(name4 | (Rn)),]
       [TWASIZE(name4 | (Rn)),]
       [STATUS(name1),]
       [PARTITIONSET(name1),]
       [PARTITIONSET_NAME(name8),]
       [TASKDATAKEY(name1),]
       [TASKDATALOC(name1),]
       [STORAGE_CLEAR(name1),]
       [SYSTEM_RUNAWAY(name1),]
       [RUNAWAY_LIMIT(name4 | (Rn)),]
       [DYNAMIC(name1),]
       [REMOTE_SYSTEM(name4),]
       [REMOTE_NAME(name8),]
       [TRAN_ROUTING_PROFILE(name8),]
       [LOCAL_QUEUING(name1),]
       [TCLASS(name1),[TCLASS_NAME(name8),]]
       [DTIMEOUT(name4 | (Rn)),]
       [INDOUBT(name1),]
       [RESTART(name1),]
       [SPURGE(name1),]
       [TPURGE(name1),]
       [DUMP(name1),]
       [TRACE(name1),]
       [SHUTDOWN(name1),]
       [RESSEC(name1),]
       [CMDSEC(name1),]
       [STORAGE_FREEZE(name1),]
       [REMOTE(name1),]
       [ISOLATE(name1),]
       [SYSTEM_ATTACH(name1),]
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

The following parameter descriptions explain briefly the possible values that can be returned on an INQUIRE_TRANDEF call. For a more detailed explanation of some of these parameters, see the corresponding parameter descriptions for the TRANSACTION resource definition in the *CICS/ESA Resource Definition Guide*.

INQ_TRANSACTION_ID(name4 | string | 'string')
specifies the transaction identifier for this transaction definition inquiry.

name4 The name of a 4-byte location that contains the name of the transaction identifier.

string A string of characters, without intervening blanks, naming the transaction identifier.

'string' A string of characters, within quotation marks, naming the transaction identifier. The string length is set to 4 by padding with blanks within the quotation marks.

TRANSACTION_ID(name4)
is the primary transaction identifier for this transaction definition inquiry.

name4 The name of a 4-byte location that contains the name of the transaction identifier.

INITIAL_PROGRAM(name8)
is the name of the initial program to be given control for the transaction.

name8 The name of an 8-byte location to receive the initial program name.

PROFILE_NAME(name8)
is the name of the profile definition that is associated with the transaction definition.

name8 The name of an 8-byte location to receive the name of the profile definition associated with the transaction definition.

TRAN_PRIORITY(name4 | (Rn))
is the transaction priority specified on the transaction definition.

name4 The name of a 4-byte location to receive the transaction priority, expressed as a binary value.

(Rn) A register to receive the transaction priority, expressed as a binary value.

TWASIZE(name4 | (Rn))
is the size of the transaction work area specified on the transaction definition.

name4 The name of a 4-byte location to receive the size of the transaction work area, expressed as a binary value.

(Rn) A register to receive the size of the transaction work area, expressed as a binary value.

STATUS(name1)
indicates, in a 1-byte location (*name1*), the status of the transaction definition.

The equated values for the transaction status are:

ENABLED The transaction definition is enabled.

DISABLED The transaction definition is disabled.

PARTITIONSET(name1)
indicates, in a 1-byte location (*name1*), the partitionset specified on the transaction definition.

The equated values for the partitionset are:

NONE There is no partitionset specified for the transaction definition.

KEEP The reserved name KEEP is specified for the partitionset, which means tasks running under this transaction definition use the application partitionset for the terminal associated with the transaction.

OWN The reserved name OWN is specified for the partitionset, which means tasks running under this transaction definition perform their own partitionset management.

NAMED The partitionset is named specifically on the transaction definition. The name is returned on the PARTITIONSET_NAME parameter.

PARTITIONSET_NAME(name8)
is the name of the partitionset defined on the transaction definition.

name8 The name of an 8-byte location that is to receive the name of the partitionset.

TASKDATAKEY(name1)
indicates, in a 1-byte location (*name1*), the storage key of task-lifetime storage for tasks associated with this transaction definition.

The equated values for the storage key are:

CICS CICS key is specified for task-lifetime storage.

USER USER key is specified for task-lifetime storage.

TASKDATALOC(name1)
indicates, in a 1-byte location (*name1*), the data location of task-lifetime storage for tasks associated with this transaction definition.

The equated values for the task data location are:

ANY Task-lifetime storage can be located above 16MB in virtual storage.

BELOW Task-lifetime storage must be located below 16MB in virtual storage.

STORAGE_CLEAR(name1)
indicates, in a 1-byte location (*name1*), whether task-lifetime storage, of tasks associated with this transaction definition, is to be cleared before it is freed by a FREEMAIN command.

The equated values for the storage-clear function are:

YES Task-lifetime storage must be cleared before it's freed.

NO Task-lifetime storage need not be cleared before it's freed.

SYSTEM_RUNAWAY(name1)
indicates, in a 1-byte location (*name1*), whether the transaction definition specifies the system default runaway-task time limit, which is specified on the ICVR system initialization parameter.

The equated values for system runaway are:

YES The transaction definition specifies the system default runaway limit.

NO The transaction is not governed by the system runaway limit.

RUNAWAY_LIMIT(name4 | (Rn))
is the runaway-task time limit specified on the transaction definition. If SYSTEM_RUNAWAY is YES, the value returned is the value defined by the ICVR system initialization parameter.

name4 The name of a 4-byte location that is to receive the task runaway limit, expressed as a binary value.

(Rn) A register to receive the task runaway limit, expressed as a binary value.

DYNAMIC(name1)
indicates, in a 1-byte location (*name1*), whether the transaction is defined for dynamic transaction routing.

The equated values for the dynamic attribute are:

YES The transaction is to be dynamically routed to a remote CICS.

NO The transaction is not to be dynamically routed.

REMOTE_SYSTEM(name4)
is the name of the remote system as specified on the transaction definition.

If the DYNAMIC parameter returns YES, REMOTE_SYSTEM returns the default name, which can be changed by the dynamic routing program.

If the DYNAMIC parameter returns NO, this is the actual remote system to which the transaction is to be routed.

name4 The name of a 4-byte location to receive the defined name of the remote system.

REMOTE_NAME(name8)
is the name by which the transaction is known in a remote system.

name8 The name of an 8-byte location to receive the transaction's remote name.

TRAN_ROUTING_PROFILE(name8)
is the name of the profile that CICS is to use to route the transaction to a remote system.

name8 The name of an 8-byte location to receive the transaction-routing profile.

LOCAL_QUEUING(name1)
indicates, in a 1-byte location (*name1*), whether a start request for this transaction is eligible to queue locally if the transaction is to be started on another system, and the remote system is not available.

The equated values local queuing are:

YES The request can be queued locally. storage.

NO The request is not to be queued locally.

TCLASS(name1)
indicates, in a 1-byte location (*name1*), whether the transaction belongs to a transaction class.

The equated values for transaction class are:

YES The transaction is a member of the transaction class named in the TCLASS_NAME parameter.

NO The transaction is not a member of a transaction class.

TCLASS_NAME(name8)
is the name of the transaction class to which the transaction belongs.

name8 The name of an 8-byte location to receive transaction class name to which the transaction belongs.

DTIMEOUT(name4)
is the deadlock time-out value for the transaction.

name4 The name of a 4-byte location that is to receive the deadlock time-out value, expressed as a binary value.

(Rn) A register to receive the deadlock time-out value, expressed as a binary value.

Note that a value of zero means that the transaction resource definition specifies DTIMOUT(NO).

INDOUBT(name1)
indicates, in a 1-byte location (*name1*), the action CICS is to take if the transaction

abends while using intercommunication, and syncpointing status is in doubt.

The equated values for indoubt are:

BACKOUT Any changes made by the transaction to recoverable resources are to be backed out.

COMMIT Any changes made by the transaction to recoverable resources are to be committed.

WAIT Any changes made to recoverable temporary storage are locked until the session is recovered.

RESTART(name1)
indicates, in a 1-byte location (*name1*), whether the transaction restart is to be considered for transaction restart.

The equated values for restart are:

YES The transaction can be restarted.

NO The transaction cannot be restarted.

SPURGE(name1)
indicates, in a 1-byte location (*name1*), whether the transaction is defined as system-purgeable.

The equated values for system-purgeable are:

YES The transaction is system-purgeable.

NO The transaction is not system-purgeable.

TPURGE(name1)
indicates, in a 1-byte location (*name1*), whether the transaction is defined as purgeable in the event of a VTAM terminal error.

The equated values for terminal-purgeable are:

YES The transaction can be purged if a terminal error occurs.

NO The transaction can not be purged if a terminal error occurs.

DUMP(name1)
indicates, in a 1-byte location (*name1*), whether CICS is to take a transaction dump if the transaction abends.

The equated values for dump are:

YES A transaction dump is required.

NO A transaction dump is not required.

TRACE(name1)
indicates, in a 1-byte location (*name1*), the level of tracing defined for the transaction

The equated values for trace are:

STANDARD
CICS standard-level trace This equates to TRACE(YES) in the TRANSACTION resource definition.

SPECIAL CICS special-level trace This is the result of special trace being set by means of an EXEC CICS SET TRANSACTION command.

SUPPRESSED
Tracing is suppressed for the transaction This equates to TRACE(NO) in the TRANSACTION resource definition.

SHUTDOWN(name1)
indicates, in a 1-byte location (*name1*), whether the transaction can be run during CICS shutdown.

The equated values for shutdown are:

ENABLED The transaction is enabled to run during CICS shutdown.

DISABLED The transaction is disabled from running during CICS shutdown.

RESSEC(name1)
indicates, in a 1-byte location (*name1*), whether resource security checking is required for the transaction. The equated values for resource security are:

YES Resource security checking is required.

NO Resource security checking is not required.

CMDSEC(name1)
indicates, in a 1-byte location (*name1*), whether command security checking is required for the transaction. The equated values for command security are:

YES Command security checking is required.

NO Command security checking is not required.

STORAGE_FREEZE(name1 | (Rn))
indicates, in a 1-byte location (*name1*), whether storage freeze is defined for the transaction by means of the STGFRZ option on the CICS-supplied field engineering transaction, CSFE.

The equated values for storage freeze are:

YES Storage that is normally freed during the running of a transaction is frozen.

NO Storage is freed normally during the running of the transaction.

REMOTE(name1)
indicates, in a 1-byte location (*name1*), whether the transaction is defined as remote.

The equated values for a remote transaction are:

YES The transaction is a remote transaction.

NO The transaction is not a remote transaction.

ISOLATE(name1)
indicates, in a 1-byte location (*name1*), whether transaction isolation is required for the transaction's task-lifetime user-key storage.

The equated values for isolate are:

YES Transaction isolation is required for task-lifetime user-key storage.

NO Transaction isolation is not required for task-lifetime user-key storage.

SYSTEM_ATTACH(name1)
indicates, in a 1-byte location (*name1*), whether the tasks attached with this tranid are always to be attached as system tasks.

The equated values for isolate are:

YES A system task is being attached for this transaction.

NO A user task is being attached for this transaction.

RESPONSE and REASON values for INQUIRE_TRANDEF:

RESPONSE	*REASON*
OK	None
EXCEPTION	UNKNOWN_TRANSACTION_ID
INVALID	None
DISASTER	LOGIC_ERROR
PURGED	None

The INQUIRE_TRANSACTION function

The INQUIRE_TRANSACTION function is provided on the DFHXMIQX macro call. Its purpose is to allow you to obtain information about a transaction that is attached (task). In general, this function call is equivalent to the EXEC CICS INQUIRE TASK command, with some minor differences.

The syntax of the call is shown as follows:

INQUIRE_TRANSACTION

```
DFHXMIQX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_TRANSACTION),
       [TRANSACTION_TOKEN(name8)],]
       [OUT,
       [CICS_UOW_ID(name8),]
       [CMDSEC(name1),]
       [DTIMEOUT(name4 | (Rn)),]
       [DUMP(name1),]
       [DYNAMIC(name1),]
       [FACILITY_NAME(name4),]
       [FACILITY_TYPE(name1),]
       [INDOUBT(name1),]
       [INITIAL_PROGRAM(name8),]
       [ISOLATE(name1),]
       [LOCAL_QUEUING(name1),]
       [NETNAME(name8),]
       [ORIGINAL_TRANSACTION_ID(name4),]
       [OUT_TRANSACTION_TOKEN(name8),]
       [PROFILE_NAME(name8),]
       [REMOTE(name1),]
       [REMOTE_NAME(name8),]
       [REMOTE_SYSTEM(name4),]
       [RESOURCE_NAME(name8),]
       [RESOURCE_TYPE(name8),]
       [RESSEC(name1),]
       [RESTART(name1),]
       [RESTART_COUNT(name2 | (Rn)),]
       [RUNAWAY_LIMIT(name4 | (Rn)),]
       [SPURGE(name1),]
       [START_CODE(name1),]
       [STATUS(name1),]
       [STORAGE_CLEAR(name1),]
       [SUSPEND_TIME(name4 | (Rn)),]
       [SYSTEM_TRANSACTION(name1),]
       [TASK_PRIORITY(name1),]
       [TASKDATAKEY(name1),]
       [TASKDATALOC(name1),]
       [TCLASS(name1),[TCLASS_NAME(name8),]]
       [TPURGE(name1),]
       [TRACE(name1),]
       [TRAN_PRIORITY(name1),]
       [TRAN_ROUTING_PROFILE(name8),]
       [TRANNUM(name4 | string | 'string'),]
       [TRANSACTION_ID(name4),]
       [TWASIZE(name4),]
       [USERID(name8),]
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

The descriptions of the following parameters are the same as the corresponding parameters on the INQUIRE_TRANDEF function call.

```
CMDSEC            RESSEC
DTIMEOUT          RESTART
DUMP              RUNAWAY_LIMIT
DYNAMIC           SPURGE
INDOUBT           STATUS
INITIAL_PROGRAM   STORAGE_CLEAR
ISOLATE           TASKDATAKEY
LOCAL_QUEUING     TASKDATALOC
PROFILE_NAME      TCLASS
REMOTE            TRAN_ROUTING_PROFILE
REMOTE_NAME       TRANSACTION_ID
REMOTE_SYSTEM     TWASIZE
```

The parameter descriptions that follow explain briefly the possible values that can be returned on an INQUIRE_TRANSACTION call. For a more detailed explanation of these parameters, see the corresponding parameter descriptions for the TRANSACTION resource definition in the *CICS/ESA Resource Definition Guide*.

TRANSACTION_TOKEN(name8)
specifies the transaction token for the task being inquired upon. This parameter is optional, and if omitted, the current task is assumed.

If you issue this call within an XXMATT global user exit program, the current task may be a CICS system task. To inquire on the user task for which XXMATT is invoked, you must specify the transaction token passed on the XXMATT exit-specific parameter list.

name8 The name of an 8-byte location that contains the transaction token.

CICS_UOW_ID(name8)
is the CICS unit of work identifier for the task.

name8 The name of an 8-byte location to receive the unit of work id.

FACILITY_NAME(name4)
is the name of the principal facility associated with the task.

name4 The name of a 4-byte location to receive the name of the principal facility.

FACILITY_TYPE(name1)
indicates, in a 1-byte location (*name1*), the type of principal facility associated with the task.

The equated values for the facility type are:

NONE There is no principal facility.

TERMINAL The principal facility is a terminal.

TD The principal facility is a transient data queue.

START The principal facility is an interval control element.

NETNAME(name8)
is the network name of the principal facility associated with this task.

name8 The name of an 8-byte location to receive the network name.

ORIGINAL_TRANSACTION_ID(name4)
is the transaction id that was used to attach the transaction. For example, if an alias was used at a terminal, this field returns the alias.

name4 The name of a 4-byte location to receive the name of the original transaction identifier.

OUT_TRANSACTION_TOKEN(name8)
is the token that represents the task.

name8 The name of an 8-byte location to receive the transaction token for the task.

RESOURCE_NAME(name8)
is the name of a resource that the (suspended) transaction waiting for.

name8 The name of an 8-byte location to receive the name of the resource on which the transaction is waiting.

RESOURCE_TYPE(name8)
is the type of resource that the (suspended) transaction waiting for.

name8 The name of an 8-byte location to receive the type of resource on which the transaction is waiting.

RESTART_COUNT(name2 | (Rn))
is the number of times this instance of the transaction has been restarted.

name2 The name of a 2-byte location to receive the number of times the transaction has been restarted, expressed as a half-word binary value.

(Rn) A register to receive the number of times the transaction has been restarted, expressed as a half-word binary value.

START_CODE(name1)
indicates, in a 1-byte location (*name1*), how the task was started:

The equate values for the start codes are:

C A CICS internal attach.
T A terminal input attach.
TT A permanent transaction terminal attach.
QD A transient data trigger level attach.
S A START command without any data.
SD A START command with data.
SZ A front end programming interface (FEPI) attach.
DF The start code isn't yet known—to be set later.

SUSPEND_TIME(name4 | (Rn))
is the length of time that the task has been in its current suspended state.

name4 The name of a 4-byte location to receive the number of seconds, rounded down, the task has been suspended, expressed as a binary value.

(Rn) A register to receive the number of seconds, rounded down, the task has been suspended, expressed as a binary value.

SYSTEM_TRANSACTION(name1)
indicates, in a 1-byte location (*name1*), whether the task is CICS system task.

The equated values for the facility type are:

YES The task is a CICS system task.

NO The task is not a CICS system task.

TASK_PRIORITY(name1)
is the combined task priority, which is the sum of the priorities defined for the terminal, transaction, and operator.

name1 The name of a 1-byte location to receive the task priority, expressed as a binary number.

TRANNUM(name4)
is the task number of the transaction.

name4 The name of a 4-byte location to receive the task number.

USERID(name8)
is the userid associated with this task.

name8 The name of an 8-byte location to receive the userid.

RESPONSE and REASON values for INQUIRE_TRANSACTION:

RESPONSE	*REASON*
OK	None
DISASTER	ABEND
	LOOP
INVALID	None
EXCEPTION	NO_TRANSACTION_
	ENVIRONMENT
	BUFFER_TOO_SMALL
	INVALID_TRANSACTION_TOKEN
KERNERROR	None

The SET_TRANSACTION function

The SET_TRANSACTION function is provided on the DFHXMIQX macro call. Its purpose is to allow you to change the task priority and transaction class of the current task.

Note that you can use this function to change the TCLASS_NAME only when it is invoked from an XXMATT global user exit program.

The syntax of the call is shown as follows:

SET_TRANSACTION

```
DFHXMIQX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(SET_TRANSACTION),
       [TRANSACTION_TOKEN(name8),]
       [TASK_PRIORITY(name4),]
       [TCLASS_NAME(name8),]
       [OUT,
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

TRANSACTION_TOKEN(name8)
specifies the transaction token that represents the task being modified. If you omit this parameter, the call defaults to the current task.

name8 The name of an 8-byte location that contains the transaction token.

TASK_PRIORITY(name4)
specifies the new task priority being set for the task identified by TRANSACTION_TOKEN.

name4 The name of a 4-byte location that contains the new task priority number, expressed as a binary value.

TCLASS_NAME(name8)
specifies the new transaction class name that you want to associate this task with. To specify that the task is not to be in any transaction class, specify the special default system name DFHTCL00.

name8 The name of an 8-byte location that contains the name of the new transaction class.

RESPONSE and REASON values for SET_TRANSACTION:

RESPONSE	*REASON*
OK	None
EXCEPTION	NO_TRANSACTION_
	ENVIRONMENT
	UNKNOWN_TCLASS
	INVALID_TRANSACTION_TOKEN
DISASTER	ABEND
	LOOP
INVALID	None
KERNERROR	None

The kernel domain XPI calls are described in the following sections.

The START_PURGE_PROTECTION function:

The START_PURGE_PROTECTION function is provided on the DFHKEDSX macro call. It inhibits purge, but not force-purge, for the current task. This function can be used by all global user exit programs if they want to inhibit purge during a global user exit call.

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.

The syntax of the call is shown as follows:

START_PURGE_PROTECTION

```
DFHKEDSX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(START_PURGE_PROTECTION),]
       [OUT,
       RESPONSE (name1 | *)]
```

There are no input or output parameters on this call, only a RESPONSE.

RESPONSE values for START_PURGE_PROTECTION:

RESPONSE	*REASON*
OK	None
DISASTER	None
INVALID	None

The STOP_PURGE_PROTECTION function:

The STOP_PURGE_PROTECTION function is provided on the DFHKEDSX macro call. It is re-enables purge for the current task after purge has been suspended by a preceding START_PURGE_PROTECTION function call.

The syntax of the call is shown as follows:

STOP_PURGE_PROTECTION

```
DFHKEDSX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(STOP_PURGE_PROTECTION),]
       [OUT,
       RESPONSE (name1 | *)]
```

There are no input or output parameters on this call, only a RESPONSE.

RESPONSE values for STOP_PURGE_PROTECTION:

RESPONSE	*REASON*
OK	None
DISASTER	None
INVALID	None

Nesting purge protection calls:

Note that the START_ and STOP_PURGE_PROTECTION functions can be nested. You should ensure that, if multiple START_PURGE_PROTECTION calls are issued for a task, that the correct number of STOP_PURGE_PROTECTION calls are issued to cancel the purge protection. If you issue two starts and only one stop, purge protection is left on for the current task.

For example, for any current task, more than one global user exit program may be driven. You must design your exit programs to ensure that purge protection is correctly cancelled. An example of nesting is shown as follows:

```
XEIIN:
 EXIT_PROG1: Calls START_PURGE_PROTECTION

   XFCREQ:
    EXIT_PROG2: Calls START_PURGE_PROTECTION

   XFCREQC:
    EXIT_PROG3: Calls STOP_PURGE_PROTECTION

XEIOUT:
 EXIT_PROG4: Calls STOP_PURGE_PROTECTION
```

Changes to user-replaceable modules

The DFHRTY transaction restart program is replaced by DFHREST.

The transaction restart user-replaceable program (DFHREST) enables you to participate in the decision as to whether a transaction should be restarted or not. The default program requests restart under certain conditions; for example, in the event of a program isolation deadlock (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 general information about restarting transactions, see the *CICS/ESA Recovery and Restart Guide*.

Notes:

1. CICS invokes DFHREST only when RESTART(YES) is specified in a transaction's resource definition.
2. When transaction restart occurs, a new task is attached that invokes the initial program of the transaction. This is true even if the task abended in the second or subsequent LUW, and DFHREST requested a restart.
3. Statistics on the total number of restarts against each transaction are kept.
4. Emergency restart does not restart any tasks.
5. Making a transaction restartable involves slightly more overhead than dynamic transaction backout because more items are logged; such items are logged only on the dynamic log.
6. In some cases, the benefits of transaction restart can be obtained instead by using the SYNCPOINT ROLLBACK command. Although use of the ROLLBACK command is not usually recommended, it does keep all the executable code in the application programs (except for DFHDBP exit code). For more information about the use of the ROLLBACK option when working in an ISC or MRO environment, see the *CICS/ESA Intercommunication Guide*.

When planning to replace the default DFHREST, check to see if the logic of any of your transactions is inappropriate for restart.

- Transactions that execute as a single logical unit of work are safe. Those that execute a loop, and on each pass reading one record from a recoverable destination, updating other recoverable resources, and closing with a syncpoint, are also safe.
- There are two types of transaction that need to be modified to avoid erroneously repeating work done in the logical units of work that precede an abend:
    1. A transaction in which the first and subsequent logical units of work change different resources
    2. A transaction where the contents of the input data area are used in several logical units of work.

Conditions for invoking the transaction restart module

All the following conditions must be true for CICS to invoke the transaction restart program:

- A transaction must be terminating abnormally
- The transaction abend which caused the transaction to be terminating abnormally must have been detected before the commit point of the implicit syncpoint at the end of the transaction has been reached
- The transaction must be defined as restartable in its transaction definition
- The transaction must be related to a principal facility.

If these conditions are satisfied, CICS invokes the transaction restart program, which then decides whether or not to request that the transaction be restarted. CICS can subsequently override the decision (for example, if dynamic backout fails). Also, if the transaction restart program abends, the transaction is not restarted.

If the above conditions are not satisfied, CICS does not invoke the transaction restart program and the transaction is not restarted.

The DFHREST communications area

The CICS-supplied default transaction restart program is written in assembler and contains logic to:

- Address the communications area passed to it by CICS
- Decide whether or not to request transaction restart
- Send a message to CSMT if restart is requested
- Return control to CICS using the EXEC CICS RETURN command.

The communications area is mapped by the `XMRS_COMMAREA` DSECT, which is supplied in the DFHXMRSD copybook. The equivalent structures for C/370, COBOL, VS COBOL II, and PL/1 are contained in the copybooks DFHXMRSH, DFHXMRSO, and DFHXMRSP, respectively.

The information passed in the communications area is as follows:

XMRS_FUNCTION
Indicates, in a 1-byte field, the function code for this call to the restart program. This is always set to 1, which equates to `XMRS_TRANSACTION_RESTART`, which means that DFHREST is called to handle transaction restart.

XMRS_COMPONENT_CODE
Indicates, in a 2-byte field, the component code of the caller. This is always set to XM, which equates to `XMRS_TRANSACTION_MANAGER`. The transaction manager is the CICS component that coordinates the decision whether or not to restart a transaction.

XMRS_READ
Indicates, in a 1-byte field, whether the transaction has issued any terminal read requests, other than for initial input.

The equated values for this parameter are:

XMRS_READ_YES Means a terminal read has been performed by the transaction.

XMRS_READ_NO Means no terminal read has been performed.

XMRS_WRITE
Indicates, in a 1-byte field, whether the transaction has issued any terminal write requests.

The equated values for this parameter are:

XMRS_WRITE_YES Means a terminal write has been performed by the transaction.

XMRS_WRITE_NO Means a terminal write has not been performed by the transaction.

XMRS_SYNCPOINT
Indicates, in a 1-byte field, whether the transaction has performed any syncpoints.

The equated values for this parameter are:

XMRS_SYNCPOINT_YES Means one or more syncpoints have been performed.

XMRS_SYNCPOINT_NO Means no syncpoints have been performed.

XMRS_RESTART_COUNT
This indicates, as an unsigned, half-word binary value, the number of times the transaction has been restarted.

It is zero if the transaction has not been restarted. It is **not** the total number of restarts for the transaction definition. Rather it is the total number of restarts for transactions that are attempting, for example, to process a single piece of operator input.

XMRS_ORIGINAL_ABEND_CODE
Provides the first abend code recorded by the transaction.

XMRS_CURRENT_ABEND_CODE
Provides the current abend code. The values of the original abend code and the current abend code can be different if, for example, a transaction handles an abend and then abends later.

XMRS_RESTART
This is a 1-byte output field that the transaction restart program sets to indicate whether it wants CICS to restart the transaction.

The equated values for this field are:

XMRS_RESTART_YES Requests a restart.

XMRS_RESTART_NO Requests no restart.

The CICS-supplied default transaction restart program requests that the transaction be restarted if:

1. The transaction has not performed a terminal read (other than reading the initial input data), terminal write or syncpoint, **and**
2. The restart count is less than 20 (to limit the number of restarts), **and**
3. The current abend code is either ADCD or ADLD indicating that the transaction was abended due to a DBCTL deadlock or a program isolation deadlock, respectively.

The source of the CICS-supplied default transaction restart program, DFHREST, is supplied in assembler language only, in the CICS410.SDFHSAMP library.

The assembler copybook for mapping the communications area is in the CICS410.SDFHMAC library.

End of Product-sensitive programming interface

Change to statistics

You can issue both EXEC CICS COLLECT STATISTICS and EXEC CICS PERFORM STATISTICS RECORD commands for TRANCLASS.

New statistics and extensions of existing statistics are provided for transaction manager. See the *CICS/ESA Performance Guide* for details of all the statistics records.

Problem determination

There are changes to messages and codes, and new trace points for the transaction manager domain. For details of the trace entries, see the *CICS/ESA Diagnosis Reference*.

Changed messages

The following messages have been removed:

DFHKC0306
DFHAC2210
DFHAC2111
DFHAC2212
DFHAC2213
DFHAC2214
DFHAC2240
DFHAC2242
DFHAC2243
DFHAC2244

The following messages have been replaced:

DFHKC0101 by DFHXM0101
DFHKC0103 by DFHXM0103
DFHKC0105 by DFHXM0105
DFHBP0803 by DFHXM0301
DFHBP0804 by DFHXM0804

For new messages and codes relating to transaction manager see the XM section of the *CICS/ESA Messages and Codes* manual.

Abend codes

AED7 is discontinued.

AEDA CEDF transaction has started with invalid start code.

AEDB DFHEDFP has been passed an invalid EDFXA.

AEDC An error occurred in an EDF call to storage manager.

AEDD Attach of CEDF transaction failed. Check that the CEDF transaction is available.

AEDE A CEDF task received unexpected response from the resumption of the user task.

AEDF A CEDF task has attempted to access the user task, but the user task has gone. Action: determine the reason for the loss of the user task.

Chapter 22. Security manager domain

This chapter describes the changes in CICS/ESA 4.1 that are associated with the introduction of the new security manager domain. It covers the following topics:

- Overview
- Benefits of the security manager domain
- Requirements for the security manager domain
- Changes to CICS externals
- Problem determination.

Note: In general, this chapter assumes that the external security manager (ESM) is the IBM Resource Access Control Facility (RACF).

Overview

There are two aspects to security processing in releases of CICS before CICS/ESA 4.1:

1. The identification of the terminal users of the CICS system, and permitting them to sign on to CICS.

   For this purpose, CICS creates a signon table terminal entry (SNTTE) to represent the terminal user. CICS uses the entries in the signon table to hold non-security attributes of the user, such as the operator priority, the operator classes, and the user's national language.

2. Assignment of capabilities or authorities to users identified by SNTTEs.

   For this purpose, CICS obtains an accessor environment element (ACEE) from the external security manager (ESM), which is required whenever a user's security authorization is being queried (for example when a user's authority to access a resource is being determined).

The SNTTE and the ACEE are intimately bound together, and both are invariably bound to a TCTTE. This means that a task without a principal facility cannot acquire an SNTTE, and therefore cannot acquire an ACEE, and therefore cannot be granted authorities.

The restructure of these two functional areas of CICS creates two independent domains—one to encapsulate users' capabilities or authorities (the security manager domain) and the other to encapsulate users' attributes (the user domain). The independence of the security manager and user domains means they are no longer dependent on the presence of a principal facility or TCTTE.

The security manager domain is primarily a restructure of the function previously provided by the CICS DFHSEC macros and the associated security management modules, and is complementary to the user domain (see Chapter 23, “User domain” on page 401 for information about the restructure of the management of user data). It also provides services for the signon component, which is also restructured in this release (see Chapter 24, “Signon component” on page 407 for information about the changes in signon).

The restructure of the security function encapsulates the recording of users' capabilities or authorities in the security manager domain. The security manager domain handles all the interfaces to the external security manager.

As part of the restructure, the security function is enhanced by the addition of new function. The main changes are:

- Independence from terminals
- The addition of surrogate user checks
- Security checks for category 1 CICS-supplied transactions
- Support for PassTickets, used by the CICS front-end programming interface (see "Security facilities for FEPI" on page 480)
- API enhancements for password management
- Auditing for LU6.2 bind security
- Improved messages (see "Changes to messages and abend codes" on page 399 for details)
- Support for RACF groups
- Support for RACF Version 2 Release 1

Independence from terminals

With the introduction of the user domain for the management of user data, the CICS internal signon table of earlier releases is obsolete in CICS/ESA 4.1. This frees CICS security from its dependence on terminal entries in order to locate user data in signon table entries (SNTTEs).

As a consequence of this, the transaction manager domain, user domain, and security domain all combine to provide a security environment that is built round the transaction, and not the terminal.

Surrogate user checks

CICS uses the RACF surrogate user facility to check that a user is authorized to act for another user. You can enforce surrogate user checking for the following:

Started transactions

CICS performs surrogate users checks when you start a transaction that is not associated with a terminal.

See "Changes to the application programming interface" on page 130 for information about surrogate user checking for non-terminal started transactions.

Preset terminal security

When you install a terminal that is defined with a preset security userid, CICS checks that the userid performing the install is authorized as a surrogate user.

See "Surrogate user checking for terminals with preset security" on page 398 for more information.

CICS region default user

CICS performs a surrogate user security check against its own userid (the CICS region's userid) to ensure it is properly authorized to the default userid specified on the DFLTUSER system initialization parameter.

See “Surrogate user checking of the CICS default userid” on page 398 for more information.

Transient data trigger-level transactions

When a transient data queue is defined with a non-terminal, trigger-level transaction, CICS checks that its own userid is authorized as a surrogate user of the userid specified for the trigger-level transaction.

See “Changes to resource definition (macro)” on page 134 for information about surrogate user checking for non-terminal trigger-level transactions.

PLT post-initialization processing

If you specify a program list table on a PLTPI system initialization parameter, you can select surrogate user checking against the CICS region userid by means of the PLTPISEC system initialization parameter

See “Changes to system definition” on page 128 for information about PLTPI surrogate user checking.

Surrogate checks during CICS initialization

When surrogate user checking is active, some surrogate user checking is performed against the CICS region userid during CICS startup. These initialization checks are to ensure that the CICS region userid is properly authorized as a surrogate of the following:

- The DFLTUSER userid, when signing on the CICS default user
- The PLTPIUSR userid, at the start of PLTPI processing
- Any userids specified in the DCT.

Failure of the surrogate security check against the CICS region userid for the DFLTUSER and PLTPIUSR userids causes CICS to abend.

Failure of the surrogate security check against the CICS region userid for DCT userids causes CICS to deactivate automatic transaction initiation for the affected intrapartition queue(s).

Security checks for category 1 CICS-supplied transactions

In earlier releases, category 1 CICS-supplied transactions are exempt from any security checks. In CICS/ESA 4.1, the CICS region userid must be authorized to run its own category 1 transactions, otherwise CICS abends during system initialization. For a full list of the category 1 CICS-supplied transactions, see the *CICS/ESA CICS-RACF Security Guide*.

Auditing for LU6.2 security

If an LU6.2 bind security failure occurs, this is detected by CICS and appropriate auditing information is written to the MVS System Management Facility (SMF) data set using RACF services.

API enhancements for password management

There are two CICS API commands introduced—EXEC CICS VERIFY PASSWORD and EXEC CICS CHANGE PASSWORD—to enable you to perform password management within a CICS application program.

The VERIFY command enables an application program to verify the password of a userid, and to obtain information about the userid and password, such as the date and time the userid was last accessed. The userid you specify on the VERIFY command can be different from the one actually signed on at the terminal.

You can use the CHANGE password command to change a user's password within an application program, without requiring a signoff and signon. It can be used, for example, if the password being verified on a VERIFY command is expired.

The VERIFY and CHANGE PASSWORD commands, unlike the EXEC CICS SIGNON command, do not depend on the principal facility, and can be issued when the principal facility is an APPC session.

Support for RACF groups

CICS allows users to specify the RACF group with which they want to be associated for security checking purposes, using the group id option on the new CICS signon panel.

In earlier releases, the absence of a group id means that either:

- Users have only the authorizations of their default RACF group, or
- Users have the authorizations of all the groups of which they are members if list-of-groups checking is active, which can have significant performance implications.

Support for RACF 2.1

CICS/ESA 4.1 exploits new function in RACF Version 2 Release 1:

- If you are using RACF 2.1, you no longer need to use the CICS security rebuild commands to refresh the RACF in-store profiles.
- For MRO environments, identical signon parameters are propagated between CICS regions, to exploit RACF caching of accessor environment elements (ACEEs) in the MVS coupling facility.

  For information about this propagation of the userid signed on in a terminal-owning region, see “New signon authorization processes” on page 407.

Benefits

The security manager domain conforms to the CICS/ESA domain architecture, providing greatly improved reliability. Access to security manager domain function is through the architected domain interface only: access to the security manager domain's control blocks is not permitted. Security manager domain state data is accessible only through the documented interfaces, which are extended in this release.

The enhancements incorporated in the security manager domain restructure address a number of user requirements, the most significant of which are:

- To decouple security checks from the terminal to enable CICS to support non-terminal security (see Chapter 9, “Non-terminal security” on page 127)
- To provide security checks for the introduction of work for new userids without introducing a non-password signon function. This is achieved using the surrogate user facility of RACF, with surrogate user checks performed for the following:
  - Started transactions
  - Trigger-level transactions
  - PLT-initiated programs at CICS startup
  - Installing default userids
  - Installing terminals with preset security.
- To improve CICS security messages, to provide:
  - More information in the security violation message, DFHXS1111
  - More information in the resource classes being RACLISTed at initialization and on a security rebuild.

For RACF 2.1 users there is a major system management benefit in not needing to issue multiple PERFORM SECURITY REBUILD commands to refresh security profiles across many CICS regions. This becomes a RACF function under RACF 2.1, and as such removes a significant performance overhead from CICS regions.

Requirements for the security domain

CICS/ESA 4.1 does not support the CICS signon table macro, DFHSNT, which is obsolete. To define user attributes, you must use RACF 1.9.2 or later, or an equivalent external security manager.

If you are a RACF user, you enter user attributes in the CICS segment of the user profile in the RACF database.

See the *CICS/ESA CICS-RACF Security Guide* for information about defining CICS user attributes in the RACF database.

Changes to CICS externals

The new security manager domain introduces a number of changes to CICS externals: These are:

- Changes to system definition
- Changes to resource definition
- Changes to the application programming interface
- Changes to the system programming interface
- Changes to global user exits
- Surrogate user checking.

Changes to system definition

There are some new and changed system initialization parameters introduced for the security manager:

- CMDSEC
- RESSEC
- SPCTRXS
- STNTRXS
- XUSER

In addition to the new system initialization parameters listed above, the SEC=MIGRATE option is removed. You can specify only SEC=YES or SEC=NO.

Table 32. The DFHSIT macro parameters

	DFHSIT	[TYPE={**CSECT**|DSECT}]
		⋮
		[CMDSEC={**ASIS**|ALWAYS}
		⋮
		[RESSEC={**ASIS**|ALWAYS}
		[SEC={**YES**|~~MIGRATE~~|NO}
		[SPCTRXS={(1[,2][,3])|ALL|OFF}]
		[STNTRXS={(1],2]],3])|ALL|OFF}]
		⋮
		[,XUSER={**YES**|NO}]
		⋮
	END	DFHSITBA

CMDSEC={ASIS|ALWAYS}

Specifies whether or not you want CICS to honor the CMDSEC option specified on a transaction's resource definition.

ASIS

means that CICS honors the CMDSEC option defined in a transaction's resource definition. CICS calls its command security checking routine only when CMDSEC(YES) is specified in a transaction resource definition.

ALWAYS

means that CICS overrides the CMDSEC option, and always calls its command security checking routine to issue the appropriate call to the SAF interface.

Notes:

1. Specify ALWAYS when you want to control the use of the SPI in all your transactions. Be aware that this might incur additional overhead. The additional overhead is caused by CICS issuing the command security calls on every eligible EXEC CICS command, which are *all* the system programming interface (SPI) commands.
2. If you specify ALWAYS, command checking applies to CICS-supplied transactions such as CESN and CESF. You must authorize all users of CICS-supplied transactions to use the internal CICS resources for the transactions, otherwise you will get unexpected results in CICS-supplied transactions.

Restrictions

You can code the CMDSEC parameter in the SIT, PARM, or SYSIN only.

RESSEC={ASIS|ALWAYS}
Specifies whether or not you want CICS to honor the RESSEC option specified on a transaction's resource definition.

ASIS
means that CICS honors the RESSEC option defined in a transaction's resource definition. CICS calls its resource security checking routine only when RESSEC(YES) is specified in a transaction resource definition. This is normally a sufficient level of control, because often you will need only to control the ability to execute a transaction.

ALWAYS
means that CICS overrides the RESSEC option, and always calls its resource security checking routine to issue the appropriate call to the SAF interface.

Use this option only if you need to control or audit all accesses to CICS resources. You should be aware that using this option can significantly degrade performance.

Restrictions

You can code the RESSEC parameter in the SIT, PARM, or SYSIN only.

SEC={YES|~~MIGRATE~~|NO}
The MIGRATE option is removed in CICS/ESA 4.1.

SPCTRXS={(1[,2][,3])|ALL|OFF}
Specifies the level of special tracing for the security manager domain.

STNTRXS={(1[,2][,3])|ALL|OFF}
Specifies the level of standard tracing for the security manager domain.

XUSER={YES|NO}
Specifies whether or not CICS is to perform surrogate user checks.

YES
specifies that CICS is to perform surrogate user checking in all those situations that require such checks to be made (for example, on EXEC CICS START commands without an associated terminal). For information about the various circumstances in which CICS performs surrogate user checks, see the *CICS/ESA CICS-RACF Security Guide*.

NO
Specifies that CICS is not to perform any surrogate user checking.

Restrictions

You can code the XUSER parameter in the SIT, PARM, or SYSIN only.

Changes to resource definition

The BINDPASSWORD option on the CONNECTION resource definition is obsolete and not used by CICS/ESA 4.1, although it continues to be supported in the CSD, in compatibility mode only, by the CEDA transaction and by the DFHCSDUP utility program.

If you want to use LU6.2 bind security, you must specify BINDSECURITY(YES) on the connection definition and the appropriate profiles in the APPCLU general resource class.

Changes to the application programming interface

There some additions to the application programming interface:

- A new VERIFY PASSWORD command
- A new CHANGE PASSWORD command

EXEC CICS VERIFY PASSWORD command

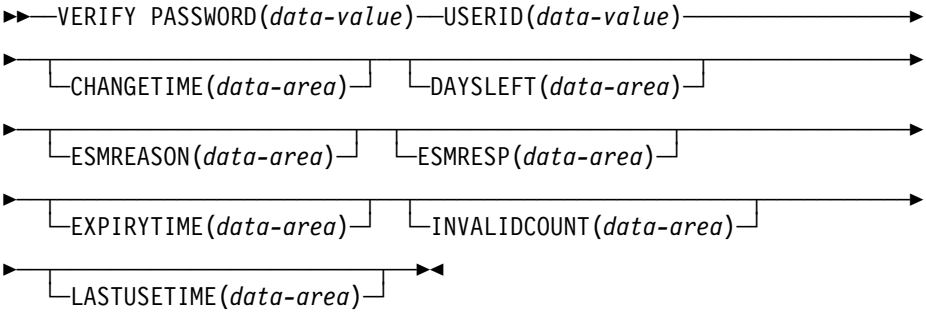
General-use programming interface

Function

You can use VERIFY PASSWORD in an application program to:

- Check a password against the information held for a userid in the RACF database without the need to perform a signon.
- Extract data about the userid and password.

Syntax



Conditions:
INVREQ, NOTAUTH, USERIDERR

In the CHANGETIME, LASTUSETIME, and EXPIRYTIME options, the time value returned is in the same format as the EXEC CICS ASKTIME command, so you can format these as date and time, in one of the formats allowed by the EXEC CICS FORMATTIME command.

Unlike the SIGNON command, the VERIFY PASSWORD command does not depend upon the principal facility, and therefore can be issued when the facility is an APPC (LU6.2) session.

VERIFY PASSWORD options

All options except USERID and PASSWORD are used to request output data.

CHANGETIME(data-area)
returns the date and time the password was last changed, in ABSTIME units.

When the external security manager is RACF, the time is shown as midnight.

DAYSLEFT(data-area)
returns the number of days from now, in a halfword binary field, until the password expires.

ESMREASON(data-area)
returns the reason code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF reason code.

ESMRESP(data-area)
returns the response code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF return code.

EXPIRYTIME(data-area)
returns the date and time the password will expire, in ABSTIME units.

When the external security manager is RACF, the time is shown as midnight.

INVALIDCOUNT(data-area)
returns, in a half-word binary field, the number of times an invalid password was entered since the last successful signon for this userid.

This count is reset whenever a successful signon occurs.

LASTUSETIME(data-area)
returns the data and time this userid was last accessed, in ABSTIME units.

PASSWORD(data-value)
specifies the password, 8 characters, that you want the ESM to check for the specified userid. The other data is not returned if the password is not valid.

USERID(data-value)
specifies the userid, 8 characters, of the user whose password is to be checked.

VERIFY PASSWORD conditions

INVREQ
occurs in any of the following situations:

- There is an unknown return code in ESMRESP from the external security manager (RESP2=13).
- The CICS external security manager interface is not initialized (RESP2=18).
- The external security manager is not responding (RESP2=29).

NOTAUTH
occurs in any of the following situations:

- The supplied password is wrong (RESP2=2). If the external security manager is RACF, the revoke count maintained by RACF is incremented.

- A new password is required (RESP2=3).
- The USERID is revoked (RESP2=19).
- The USERID's access to all its groups has been revoked (RESP2=31).

USERIDERR
occurs if the USERID is not known to the external security manager (RESP2=8).

The default action for these exception conditions is to terminate the task abnormally.

Warning: Clear password fields after use

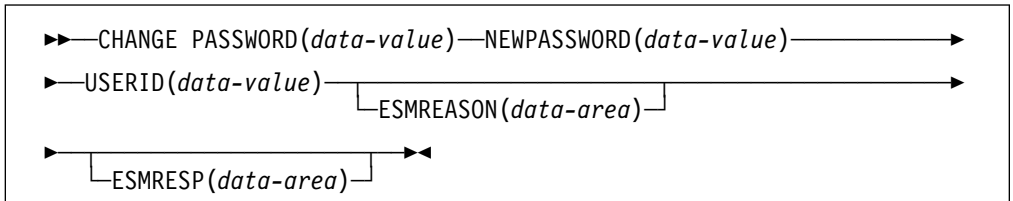
You should clear the password fields on the EXEC CICS commands that have a password option as soon as possible after use. This is to ensure that passwords are not revealed in system or transaction dumps.

EXEC CICS CHANGE PASSWORD command

Function

You can use the CHANGE PASSWORD command in an application program to change the password for a specified userid.

Syntax



Conditions:
INVREQ, NOTAUTH, USERIDERR

CHANGE PASSWORD options

Options ESMRESP and ESMREASON return the response and reason codes, if any, from the external security manager.

ESMREASON(data-area)
returns the reason code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF reason code.

ESMRESP(data-area)
returns the response code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF return code.

NEWPASSWORD(data-value)
specifies the new password, 8 characters, for the specified userid. The password is changed only if the current password is correctly specified.

PASSWORD(data-value)
specifies the current password, 8 characters, for the specified userid.

USERID(data-value)
specifies the userid, 8 characters, of the user whose password is being changed.

Unlike the SIGNON command, the CHANGE PASSWORD command does not depend upon the principal facility, and therefore can be issued when the facility is an APPC (LU6.2) session.

CHANGE PASSWORD conditions

INVREQ
occurs in any of the following situations:

- There is an unknown return code in ESMRESP from the external security manager (RESP2=13).
- The CICS external security manager interface is not initialized (RESP2=18).
- The external security manager is not responding (RESP2=29).

NOTAUTH
occurs in any of the following situations:

- The supplied password is wrong (RESP2=2). If the external security manager is RACF, the revoke count maintained by RACF is incremented.
- The new password is not acceptable (RESP2=4).
- The USERID is revoked (RESP2=19).
- The change password request failed during SECLABEL processing (RESP2=22).
- The user is revoked in the connection to the default group (RESP2=31).

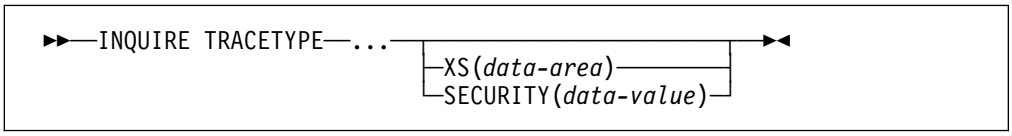
USERIDERR
occurs if the USERID is not known to the external security manager (RESP2=8).

The default action for these exception conditions is to terminate the task abnormally.

Changes to the system programming interface

The XS and SECURITY options are added to the INQUIRE and SET TRACETYPE commands.

Syntax



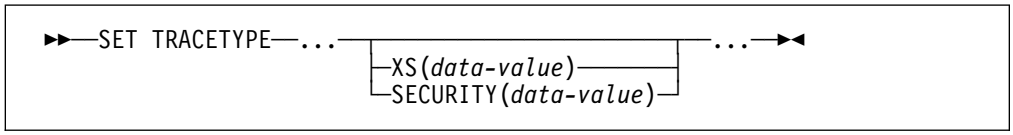
INQUIRE TRACETYPE options

XS(data-area)

returns the trace levels for the security manager domain in the form of a 32-bit string, where the bit positions correspond to the trace levels.

You can also specify SECURITY as the component identifier for the security manager domain.

Syntax



SET TRACETYPE options

XS(data-value)

specifies the trace levels for the security manager domain in the form of a 32-bit string, where the bit positions correspond to the trace levels.

You can also specify SECURITY as the component identifier for the security manager domain.

End of General-use programming interface

Changes to RACF definitions for CICS

The purpose of the RACF SURROGAT general resource class is extended for use by CICS for non-terminal security, and also for the install of terminals defined with preset security.

To enable CICS surrogate user checking, you must:

- Activate surrogate user profiles in the CICS region by specifying the system initialization parameter XUSER=YES
- Define to RACF, in SURROGAT resource class profiles, the userids of users that require the services of a surrogate user
- Authorize CICS surrogate users to the userids defined in the SURROGAT profiles.

There are two forms of surrogate class profiles that you can define for CICS surrogate user checking. The names of these SURROGAT class profiles that you define for CICS must conform to the following naming conventions:

*userid***.DFHSTART**

These profiles must be of the form *userid* concatenated with DFHSTART, where *userid* is the userid under which a started transaction is to run. They are required for started transactions that do not specify a terminal.

For an example of a DFHSTART SURROGAT resource class profile for started transactions, see “Examples of RACF surrogate user definitions” on page 397.

*userid***.DFHINSTL**

These profiles names must be of the form *userid* concatenated with DFHINSTL, where *userid* is one of the following:

- The PLT userid specified on the PLTPIUSR system initialization parameter
- The userid associated with a trigger-level transaction
- The CICS default user specified on the DFLTUSER system initialization parameter
- The userid specified for preset terminal security.

For an example of a DFHINSTL SURROGAT resource class profile, see "Examples of RACF surrogate user definitions."

Examples of RACF surrogate user definitions

You define CICS surrogate users to RACF by:

- Defining a *userid.resource_name* profile in the SURROGAT general resource class for each user that requires a surrogate user to act on their behalf. For this purpose you use the RACF RDEFINE SURROGAT command.
- Authorizing each userid that is to act as a surrogate of a user defined in a SURROGAT class profile. For this purpose you use the RACF PERMIT command.

For surrogate user checking associated with **PLTPI processing** and **transient data trigger level transactions**, you substitute the reserved name DFHINSTL for *resource_name*.

For surrogate user checking associated with **started transactions**, you substitute the reserved name DFHSTART for *resource_name*.

Surrogate users for PLT security: For PLT security checking, you must authorize the CICS region userid as a surrogate of the PLT userid defined on the PLTPIUSR system initialization parameter. This means granting the CICS region userid access to a SURROGAT resource class profile owned by the PLT userid. This is shown in the following example, where the CICS region userid is CICSHTA1, and the PLT security userid is PLTUSER:

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

PERMIT   PLTUSER.DFHINSTL  CLASS(SURROGAT) ID(CICSHTA1)  ACCESS(READ)
```

In addition to enabling PLT security by defining SURROGAT profiles, you must ensure that when PLT security is active (using the PLTPISEC system initialization parameter) you also add the PLT userid to the access lists of all the resources accessed by PLT programs. For example, if you specify PLTPISEC=RESSEC, you must ensure that the PLT userid is authorized to all the CICS resource classes for which security is active. Security checking is performed only for those resources specified on the CICS resource class system initialization parameters (XTRAN, XFCT, XCMD, and so on).

Surrogate users for started transactions: For started transactions, CICS can require as many as three levels of surrogate user. (See page 130 for details of the different surrogate users that can be required for a START command.)

For started transaction security at the first level, the userid of the transaction that issues the START command must be authorized as a surrogate of the userid specified on the START command.

For example, if a transaction running under USERID2 issues a start command for a non-terminal transaction, and the START command specifies USERID1, USERID2 must be defined to RACF as a surrogate of USERID1 (with READ authority). This is illustrated in the following RACF commands:

```
RDEFINE  SURROGAT  USERID1.DFHSTART  UACC(NONE)
                   OWNER(USERID1)

PERMIT   USERID1.DFHSTART  CLASS(SURROGAT)
         ID(USERID2)  ACCESS(READ)
```

Surrogate users for transient data trigger-level transactions: For DCT trigger-level transactions, you must authorize the CICS region userid as a surrogate of all userids defined on DCT entries. This means granting the CICS region userid access to the SURROGAT resource class profiles owned by the userids specified in the DCT. This is shown in the following example, where the CICS region userid is CICSHTA1, and the userid specified on the DFHDCT TYPE=INITIAL macro is DCTUSER:

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

PERMIT   DCTUSER.DFHINSTL  CLASS(SURROGAT) ID(CICSHTA1)  ACCESS(READ)
```

Surrogate user checking of the CICS default userid: The CICS region userid must be authorized as a surrogate of the CICS default userid specified on the DFLTUSER system initialization parameter, otherwise signing on the default user fails and CICS abends.

You must define to RACF the default userid in a SURROGAT class profile in the form *userid*.DFHINSTL.

When you have defined the DFHINSTL profile for the default userid, you must authorize the CICS region userid to the default userid's profile.

For example, if the default userid is CICSHT##, and the CICS region userid is CICSHTA1:

```
RDEFINE  SURROGAT  CICSHT##.DFHINSTL UACC(NONE) OWNER(CICSHT##)

PERMIT   CICSHT##.DFHINSTL  CLASS(SURROGAT) ID(CICSHTA1)  ACCESS(READ)
```

If you don't want to define surrogate profiles for every default userid, you can define suitable generic profiles with UACC(READ). For example, if the default userids all begin with the letters CICSH, you can define the SURROGAT class profile as follows:

```
RDEFINE  SURROGAT  CICSH*.DFHINSTL UACC(READ) OWNER(CICSHT##)
```

This enables any CICS region to use a default userid beginning with CICSH.

Surrogate user checking for terminals with preset security: For CICS terminals with preset security, you must authorize surrogates of the preset userid specified in the TERMINAL resource definition. In this case, the surrogates are the userids of anyone who uses the CEDA INSTALL command to install the preset terminal definition.

You must define to RACF the preset userid in a SURROGAT class profile in the form *userid*.DFHINSTL.

When you have defined a DFHINSTL profile for the preset terminal userid, you must authorize the userids of anyone using the CEDA INSTALL command to the terminal's preset userid profile.

For example, if the preset userid is CICSTA11, and the userid of a system programmer using the CEDA command is SYSPROG3:

```
RDEFINE  SURROGAT  CICSTA11.DFHINSTL UACC(NONE) OWNER(CICSTA11)

PERMIT   CICSHT##.DFHINSTL  CLASS(SURROGAT) ID(SYSPROG3) ACCESS(READ)
```

Problem determination

There are changes to message and codes, and new trace points, for the security manager domain. For details of the trace entries, see the *CICS/ESA Problem Determination Guide*.

Changes to messages and abend codes

The following messages are removed:

DFHXS0101 DFHXS3604
DFHXS0205 DFHXS3605
DFHXS0207 DFHXS3606

The following messages are added:

DFHXS0001	DFHXS1101	DFHXS1106	DFHXS1112	DFHXS1205
DFHXS0002	DFHXS1102	DFHXS1107	DFHXS1113	DFHXS1211
DFHXS0004	DFHXS1103	DFHXS1108	DFHXS1201	DFHXS1213
DFHXS0006	DFHXS1104	DFHXS1109	DFHXS1202	DFHXS1214
DFHXS1100	DFHXS1105	DFHXS1110	DFHXS1203	DFHXS1215

Message DFHXS0100 is replaced by DFHXS1111.

The following abend codes are no longer issued:

AXSA
AXSB
AXSC

Chapter 23. User domain

This chapter describes the user domain introduced in CICS/ESA 4.1. It covers the following topics:

- Overview
- Benefits of the user domain
- Requirements for the user domain
- Changes to CICS externals
- Problem determination.

Note: In general, this chapter assumes that the external security manager (ESM) is the IBM Resource Access Control Facility (RACF). Where function is provided specifically for non-RACF ESMs, this is made clear in the text.

Overview

The user domain is primarily a restructure of the function previously provided by the CICS signon facility, and is complementary to the security manager domain (see Chapter 22, “Security manager domain” on page 385 for information about the restructure of CICS security).

The main features of the user domain are as follows:

- The USER domain is responsible for identifying users and recording their non-security attributes. It supplies a “user token” to represent each user that signs on.
- The user domain is *not* responsible for the long-term saving of user tokens—this function passes to the signon component of CICS terminal control (see Chapter 24, “Signon component” on page 407). Thus the function of mapping security to the principal facility is eliminated from the user domain.

  The user token is also used by non-terminal facility managers, such as interval control and transient data.
- The user domain restructure incorporates a number of enhancements, many of which satisfy long-standing customer requirements.

The various enhancements are described in the following sections.

Controlling multiple signon

In earlier releases more than one terminal user can sign on to CICS using the same userid and password. In CICS/ESA 4.1 you can control multiple signon by means of a new system initialization parameter—SNSCOPE (see Table 33 on page 403 for details).

More reuse of userid entries

CICS reuses its internal representation of a user under more circumstances, within the time interval specified by the new USRDELAY system initialization parameter. This reduces the number of calls to RACF and improves performance, especially when signing on link userids.

Increased terminal timeout period

The maximum timeout period for an idle terminal is increased from 60 minutes to 99 hours and 59 minutes. You specify the timeout period in the terminal user's CICS segment in the RACF database.

Extension to the system programming interface

You can extract the default userid, using the EXEC CICS INQUIRE SYSTEM command.

Benefits of the user domain

The user domain conforms to the CICS/ESA domain architecture, providing greatly improved reliability. Access to user domain function is through the architected domain interface only: access to the user domain's control blocks is not permitted. User domain state data is accessible only through the documented interfaces, which are extended in this release.

The enhancements incorporated in the user domain restructure address a number of user requirements, the most significant of which are:

- To control multiple signons by the same userid
- To increase idle time before timeout
- To allow RACF group name at signon.

There is also benefit from the optimization of signon of SECURITYNAME userids because of userid sharing within the USRDELAY interval.

Requirements for the user domain

CICS/ESA 4.1 does not support the CICS signon table macro, DFHSNT, which is obsolete. To define user attributes, you must use RACF 1.9.2 or later, or an equivalent external security manager.

If you are a RACF user, you enter user attributes in the CICS segment of the user profile in the RACF database.

See the *CICS/ESA CICS-RACF Security Guide* for information about defining CICS user attributes in the RACF database.

Changes to CICS externals

The user domain introduces a number of changes to CICS externals in CICS/ESA 4.1. These are:

- Changes to system definition
- Changes to the system programming interface
- Changes to statistics.

Changes to system definition

There are some new system initialization parameters for the user domain, shown in Table 33.

Table 33. The DFHSIT macro parameters

	DFHSIT	[TYPE={**CSECT**|DSECT}]
		⋮
		[,SNSCOPE={**NONE**|CICS|MVSIMAGE|SYSPLEX}]
		⋮
		[,SPCTRUS={(1[,2][,3])|ALL|OFF}]
		[,STNTRUS={(1],2]],3])|ALL|OFF}]
		[,USRDELAY={**30**|number}]
		⋮
	END	DFHSITBA

SNSCOPE={NONE|CICS|MVSIMAGE|SYSPLEX}

Specifies whether or not a userid can be signed on to CICS more than once, within the scope of:

- A single CICS region
- A single MVS image
- A sysplex.

NONE

Each userid can be used to sign on for any number of sessions on any CICS region. This is the compatibility option, providing the same signon scope as in releases of CICS before CICS/ESA 4.1.

CICS

Each userid can be signed on once only in the same CICS region. A signon request is rejected if the userid is already signed on to the same CICS region. However, the userid can be used to signon to another CICS region in the same, or another, MVS image.

MVSIMAGE

Each userid can be signed on once only, and to only one of the set of CICS regions in the same MVS image that also specify SNSCOPE=MVSIMAGE. A signon request is rejected if the user is already signed on to another CICS region in the same MVS image.

SYSPLEX

Each userid can be signed on once only, and to only one of the set of CICS regions within an MVS sysplex that also specify SNSCOPE=SYSPLEX. A signon is rejected if the user is already signed on to another CICS region in the same MVS sysplex.

The signon scope (if specified) applies to all userids signing on by an explicit signon request (for example, by an EXEC CICS SIGNON command or the CESN transaction). SNSCOPE is restricted to users signing on at local terminals, or signing on after using the CRTE transaction to connect to another system.

Signon scope specified by SNSCOPE *does not* apply to:

- Non-terminal users.
- The CICS default userid, specified by the DFLTUSER system initialization parameter.
- Preset userids, specified in the USERID option of the DEFINE TERMINAL command.
- Userids for remote users, received in attach headers.
- Userids for link security. For information about which userid is used for link security on a specific connection, see the *CICS/ESA CICS-RACF Security Guide*.
- The userid specified on the PLTPIUSR system initialization parameter.
- The CICS region userid.

Restrictions

You can code the SNSCOPE parameter in the SIT, PARM, or SYSIN only.

SPCTRUS={(1[,2][,3])|ALL|OFF}
The SPCTRUS parameter specifies the level of special tracing for the user domain. The values have the same meaning as the parameters for existing SPCTRxx system initialization parameters.

STNTRUS={(1[,2][,3])|ALL|OFF}
The STNTRUS parameter specifies the level of standard tracing for the user domain. The values have the same meaning as the parameters for existing STNTRxx system initialization parameters.

USRDELAY={30|number}
Specify the maximum time, in the range 0 through 10080 minutes (up to 7 days), that an eligible userid and its associated attributes are to be retained in the user table if the userid is unused. An entry in the user table for a userid that is retained during the delay period can be reused.

The userids eligible for reuse within the USRDELAY period are any that are:

- Received from remote systems
- Specified on SECURITYNAME in CONNECTION definitions
- Specified on USERID in SESSIONS definitions
- Specified on USERID on DFHDCT TYPE=INTRA definitions
- Specified on USERID on START commands.

Within the USRDELAY period, a userid in any one of these categories can be reused in one of the other categories, provided the request for reuse is qualified with the same qualifiers. If a userid is qualified by different group id, APPLID, or terminal id, a retained entry is not reused.

If a userid is unused for more than the USRDELAY limit, it is removed from the system, and the message DFHUS0200 is issued. You can suppress this message in an XMEOUT global user exit program. If you specify USRDELAY=0, all eligible userids are deleted immediately after use, and the message DFHUS0200 is not issued.

When running a remote transaction, a userid remains signed-on to the remote CICS region (after the conversation associated with the first attach request is

complete) until the delay specified by USRDELAY has elapsed since the last transaction associated with the attach request for the userid has completed. When this event occurs, the userid is removed from the remote CICS region.

For more information about the use of USRDELAY, see the *CICS/ESA Performance Guide*.

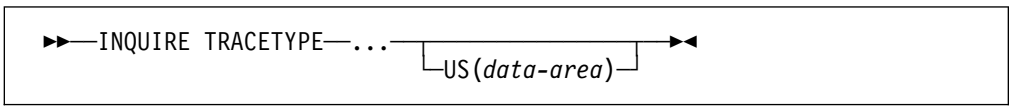
Changes to the system programming interface

General-use programming interface

The user domain component code is added to the INQUIRE and SET TRACETYPE commands.

The US component identifier for the user domain is added to the INQUIRE TRACETYPE command.

Syntax



INQUIRE TRACETYPE options

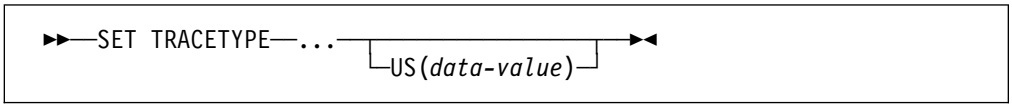
US(data-area)

returns the trace levels for the user domain in the form of a 32-bit string, where the bit positions correspond to the trace levels.

You can also specify USER as the component identifier for the user domain.

The US component identifier for the user domain is added to the SET TRACETYPE command.

Syntax



SET TRACETYPE options

US(data-value)

specifies the trace levels for the user domain in the form of a 32-bit string, where the bit positions correspond to the trace levels.

You can also specify USER as the component identifier for the user domain.

End of General-use programming interface

Changes to statistics

CICS/ESA 4.1 produces new statistics to help you select an optimum value for the USRDELAY system initialization parameter. These statistics record, among other data, the number of times a userid is timed out by exceeding the USRDELAY value, and the mean time until a userid is reused before being timed out. For more information about these statistics, see the *CICS/ESA Performance Guide*.

Problem determination

There are changes to message and codes, and new trace points, for the user domain. For details of the trace entries, see the *CICS/ESA Problem Determination Guide*.

Changes to messages

The following messages are added:

DFHUS0001
DFHUS0002
DFHUS0004
DFHUS0006
DFHUS0050
DFHUS0150
DFHUS0200

Chapter 24. Signon component

This chapter describes the changes to CICS signon, a component of CICS terminal control, which resides in the AP domain. It covers the following topics:

- Overview
- Changes to CICS externals
- Problem determination.

Note: In general, this chapter assumes that the external security manager (ESM) is the IBM Resource Access Control Facility (RACF). Where function is provided specifically for non-RACF ESMs, this is made clear in the text.

Overview

Signon is changed as a result of the security changes introduced by the restructure of the security manager and user domains.

New signon authorization processes

In earlier releases, CICS passes either its generic or specific APPLID to RACF when verifying a user's signon. This enables RACF, in addition to password checking, to also check that the user is authorized to signon to that CICS region.

There are two changes in CICS/ESA 4.1 that affect this process:

1. When signing on users in the terminal-owning region, CICS passes to RACF one of the following names as the CICS APPL name:
   - The VTAM generic resources name if GRNAME is specified as a system initialization parameter
   - The generic APPLID if one is specified on the APPLID system initialization parameter
   - The specific APPLID if only one is specified on the system initialization parameter.

   The effect of this change is that you need define only one APPL profile name in the RACF database for all the CICS regions that are members of the same VTAM generic resources name. All signon verifications in a CICSplex, where all the terminal-owning regions have the same VTAM generic resources name, are made against the same APPL profile.
2. CICS passes the APPL name used in the signon process, and the NETNAME, across all MRO links (for example, from TOR to AOR, and from AOR to FOR). When signing-on the user in application-owning region and file-owning regions, where the connection definition specifies ATTACHSEC=IDENTIFY, CICS passes the terminal-owning region's APPLID and NETNAME to RACF.

   There are several benefits from this. It enables RACF 2.1 to reuse original terminal-owning region signon information, which is cached in VLF, when CICS is signing on the user in the application-owning region. This gives a significant improvement in performance. It also reduces the number of APPL profiles you need to maintain in the RACF database, saving on security administration. Finally, it prevents users signing on directly to an application-owning region,

because the terminal-owning region APPLs are the only ones to which they are authorized.

Timeout transaction

Using the GNTRAN system initialization parameter, you can specify the transaction that you want CICS to run at a terminal that has been timed-out

GNTRAN allows you to specify your own transaction which, instead of signing off the timed-out terminal, can perform additional functions, such as locking the terminal, or prompting for a password to allow the user to remain signed-on.

+ **APAR PN89036**

+ Text for APAR PN89036 added 10 Feb 1997.

+ NO is the default for GNTRAN, which means that CICS does not invoke any
+ special transaction when the terminal timeout expires. Instead, CICS implements
+ the actions specified on the SIGNOFF attribute of the TYPETERM resource
+ definition.

CICS-supplied signon transaction enhanced

The CICS-supplied signon transaction, CESN, is enhanced to:

- Allow users to override their default RACF group name
- Use the standard three-character IBM national language code
- Maintain a previous signon until the new user data is entered
- Include the good morning message on the CESN panel
- Use some element of common user access (CUA) design for the CESN panel

See page 414 for details of the new CESN signon panel.

Improved messages

The messages associated with signon and signoff are improved in CICS/ESA 4.1 in response to user requirements. These messages use NETNAME, where appropriate, in place of TERMID to identify the terminal.

Early verification routine for security exits

| The signon process is changed to allow the external security manager products
| provided by an independent software vendor (ISV) to gain control early in the
| signon process. The change enables these products to provide an early verification
| routine, which CICS can invoke via the MVS SAF interface, as part of the CICS
| signon process.

These early verification routines can use CICS application programming interface commands, such as EXEC CICS SEND and EXEC CICS RECEIVE, to enter into a dialog with terminal users during the signon process. CICS provides a special EXEC interface environment for these early verification routines through a new EXEC interface stub.

New global user exit points

There are two global user exit points in the signon component—XSNON and XSNOFF.

Changes to CICS externals

The signon component introduces a number of changes to CICS externals in CICS/ESA 4.1. These are:

- Changes to system definition
- Changes to the application programming interface
- Changes to CICS-supplied transactions
- Changes to security exits interface.

Changes to system definition

There is a new system initialization parameter, GNTRAN, for signon, as shown in Table 34.

Table 34. The DFHSIT macro parameters

	DFHSIT	[TYPE={**CESCT**|DSECT}]
		⋮
+		[,GNTRAN={**NO**|transaction-id}.]
		⋮
	END	DFHSITBA

GNTRAN={CESF|transaction_id}

Specifies the transaction that you want CICS to invoke when a user's terminal-timeout period expires.

CESF

The default, CESF, is the basic CICS signoff transaction without any options. This transaction attempts to sign off a terminal, subject to the SIGNOFF attribute of the TYPETERM resource definition for that terminal.

transaction_id

The name of an alternative timeout transaction to signoff the user at the timed-out terminal. Specifying your own transaction allows you to specify functions in addition to, or instead of, signoff. For example, your own transaction could issue a prompt for the terminal user's password, and allow the session to continue if the correct password is entered.

Notes:

1. When either the default CESF transaction, or your own transaction, attempts to sign off a terminal, the result is subject to the SIGNOFF attribute of the TYPETERM resource definition for the terminal, as follows:

SIGNOFF	Effect
YES	The terminal is signed off, but not logged off.
NO	The terminal remains signed on and logged on.
LOGOFF	The terminal is both signed off and logged off.

2. If you use security on your CICS region, you should not specify CESN on the GNTRAN system initialization parameter because the CESN transaction does not sign off the signed-on user until a valid or an invalid attempt to sign on again is made.

Changes to the application programming interface

The enhanced signon process provided by the user domain adds five options to the SIGNON command of the CICS application programming interface.

EXEC CICS SIGNON command

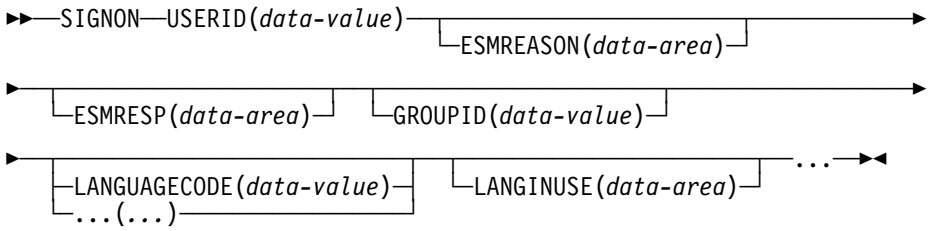
General-use programming interface

The ESMREASON, ESMRESP, GROUPID, LANGUAGECODE, and LANGINUSE options are added to the SIGNON command.

Function

Sign on to the CICS region the specified userid. To succeed, the signon request must be accompanied by a valid password that can be verified by an external security manager, such as RACF.

Syntax



Conditions:
INVREQ, NOTAUTH, USERIDERR

The LANGUAGECODE and LANGINUSE options, which use 3-character codes, are alternatives to NATLANG and NATLANGINUSE, which use a 1-character code. The 1-character CICS codes and their corresponding 3-character IBM language codes are shown in Table 35. In this table, the codes under **IBM code** are codes that you can specify in LANGUAGECODE. The codes under **Suffix** are codes that you can specify in NATLANG, and also represent the language suffix used for message domain message tables.

Table 35 (Page 1 of 2). CICS language suffixes

Suffix	IBM Code	Language name
A	ENG	Alternative, or UK, English
B	PTB	Brazilian Portuguese
C	CHS	Simplified Chinese
D	DAN	Danish
E	ENU	US English
F	FRA	French
G	DEU	German
H	KOR	Korean

Table 35 (Page 2 of 2). CICS language suffixes

Suffix	IBM Code	Language name
I	ITA	Italian
J	ISL	Icelandic
K	JPN	Japanese
L	BGR	Bulgarian
M	MKD	Macedonian
N	NOR	Norwegian
O	ELL	Greek
P	PTG	Portuguese
Q	ARA	Arabic
R	RUS	Russian
S	ESP	Spanish
T	CHT	Traditional Chinese
U	UKR	Ukrainian
V	SVE	Swedish
W	FIN	Finnish
X	HEB	Hebrew
Y	SHC	Serbo-Croatian (Cyrillic)
Z	THA	Thai
1	BEL	Byelorussian
2	CSY	Czech
3	HRV	Croatian
4	HUN	Hungarian
5	PLK	Polish
6	ROM	Romanian
7	SHL	Serbo-Croatian (Latin)
8	TRK	Turkish
9	NLD	Dutch

Note: CICS does not support fully any of these language codes except ENU and JPN. Other languages are supported only if you provide the appropriate language tables using the CICS message editing utility. For information about editing messages, see "Message editing utility program" on page 537.

SIGNON options

ESMREASON(data-area)

returns the reason code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF reason code.

ESMRESP(data-area)

returns the response code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF return code.

GROUPID(data-value)

assigns, to a RACF user group, the user that is being signed on. This overrides, for this session only, the default group name specified for the user in the RACF database.

LANGUAGECODE(data-value)

specifies the national language that the user being signed on wants CICS to use. You specify the language as a standard three-character IBM code. This

is an alternative to the one-character code that you specify on the NATLANG option.

See Table 35 on page 410 for the language codes you can use.

LANGINUSE(data-area)
the LANGINUSE option allows an application program to receive the national language chosen by the signon process. The language is identified as a standard three-character IBM code, instead of the one-character code used by NATLANGINUSE. It is an alternative to the existing NATLANGINUSE option.

End of General-use programming interface

Changes to global user exits

There are no changes to existing global user exits as a result of the restructure of security, but two new global user exit points are added to the terminal control signon component. These are XSNON and XSNOFF.

Product-sensitive programming interface

Signon exits XSNON and XSNOFF

Exit XSNON is invoked after a terminal user performs a signon, (whether successful or not), and exit XSNOFF is invoked after a terminal user performs a signoff (whether successful or not). XSNON and XSNOFF cannot make any security decisions; they are merely provided as a means of tracking users signing on and off a CICS system.

The activities that drive the exits are:

- The invocation of an EXEC CICS SIGNON command for a terminal, including signon to a remote region in a CRTE session.
- The invocation of an EXEC CICS SIGNOFF command for a terminal.
- The issue of a CANCEL to terminate a CRTE session, if the user has signed on to the remote region in the CRTE session.

Exit XSNON

	Invoked when a user signs on.	
Exit-specific parameters	**UEPUSRID**	Address of the terminal userid.
	UEPUSRLN	Address of the terminal userid length.
	UEPGRPID	Address of the group ID. If the signon was successful, the group ID is that which the user is associated with in this signon session. If the signon was unsuccessful, it is that specified by the user when he or she tried to sign on.
	UEPGRPLN	Address of the group ID length.
	UEPNETN	Address of the terminal's netname.
	UEPTRMID	Address of the terminal id.
	UEPTCTUA	Address of the TCT user area.
	UEPTCTUL	Address of the TCT user area length.

Exit XSNON (continued)

Exit-specific parameters (continued)	**UEPTRMTY**	Address of the terminal-type byte.
	UEPSNFLG	Address of a 2-byte field containing flags: **UEPSNOK** Signon was successful **UEPSNFL** Signon failed
Return codes	**UERCNORM**	Continue processing.
	UERCPURG	Task purged during XPI call.
XPI calls	All can be used.	

For the XSNOFF exit to be driven for a surrogate terminal, the terminal must not have preset security, and must be signed on (that is, TCTTESTA is set).

Exit XSNOFF

	Invoked when a user signs off.	
Exit-specific parameters	**UEPUSRID**	Address of the terminal userid.
	UEPUSRLN	Address of the terminal userid length.
	UEPGRPID	Address of the group ID.
	UEPGRPLN	Address of the group ID length.
	UEPNETN	Address of the terminal's netname.
	UEPTRMID	Address of the terminal id.
	UEPTCTUA	Address of the TCT user area.
	UEPTCTUL	Address of the TCT user area length.
	UEPTRMTY	Address of the terminal-type byte.
	UEPSNFLG	Address of a 2-byte field containing flags: **UEPSNOK** Signoff was successful **UEPSNFL** Signoff failed **UEPSNNML** Normal signoff **UEPSNTIM** Timeout signoff.
Return codes	**UERCNORM**	Continue processing.
	UERCPURG	Task purged during XPI call.
XPI calls	All can be used.	

End of Product-sensitive programming interface

Changes to CICS-supplied transactions

The CICS signon transaction, CESN, is changed to enable the user signing on to specify a group identifier (groupid), and to use the IBM 3-character language code in place of the CICS single character code. The CESN panel is also changed to include CICS "Good Morning" message as defined by the GMTEXT system initialization parameter, and to conform to common user access (CUA) standards.

Changes to CESN invocation

There is also an important change to the way CICS processes the CESN request. If you are already signed on to a CICS region, and invoke the CESN panel to sign on with a different userid, CICS does not sign off your existing session until you send the completed CESN panel. CICS signs off your existing session only when it receives the CESN panel, regardless of whether or not your new signon attempt is successful.

In earlier releases, CICS signs off the existing userid in session at the terminal as soon as you *invoke* the CESN transaction, and *before* displaying the signon panel.

Examples of the new CESN signon panel

The new CESN signon panel is shown in Figure 25 on page 415. You must now specify the language at signon using the standard three-character IBM code.

```
                    Signon for CICS/ESA Release 4.1.0        APPLID CICSHTA1

 . . . . . .  These four lines display the CICS good morning . . . . . .
 . . . . . .  message as specified on the GMTEXT system  . . . . . .
 . . . . . .  . . . .  initialization parameter  . . . . . .
 . . . . . .  . . . . . . . . . . . . . . . . . . . . . . . . . . . . .


Type your userid and password, then press ENTER:

        Userid . . . . ________    Groupid ________
        Password . . . ________
        Language . . . ___

    New Password . . . ________


DFHCE3520 Please type your userid.
F3=Exit
```

Figure 25. Example of the CESN signon panel

After successful sign-on, CICS displays message DFHCE3549, as shown in the following example where American English is the specified language:

```
DFHCE3549 Sign-on is complete (Language ENU).
```

The only change in this message is that the language code displayed is the 3-character IBM language code.

The panel for 40 x 12 size screens is shown below. In this case, there is no space for the release number or the good morning message.

```
Signon for CICS/ESA  APPLID CICSHTA1


    Userid   . . . . ________
    Groupid  . . . . ________
    Password . . . . ________
    Language . . . . ___
    New Password . . ________


DFHCE3520 Please type your userid.
F3=Exit
```

The line-command form of CESN

The line command form of the CESN transaction has been changed as follows:

- The GROUPID option is added
- The LANGUAGE option must now be specified as the standard three-character IBM code.

The full syntax is:

```
CESN USERID=uuuuuuuu[,PS=pppppppp][,NEWPS=pppppppp]
   [,GROUPID=gggggggg] [,LANGUAGE=lll]
```

Changes to the security exits interface

Product-sensitive programming interface

There are changes to the way CICS invokes the MVS system authorization facility (SAF) in CICS/ESA 4.1. These changes are made to enable security management products, provided by independent software vendors, to receive control at an early stage in the signon process. This enables ESMs provided by independent software vendors to provide an early verification routine that runs as part of the CICS signon process.

Early verification processing

The CICS signon routine invokes the SAF interface, using the RACROUTE REQUEST=VERIFY macro with the ENVIR=VERIFY option in problem-program state. Invoking this version of the macro should have no effect if the ESM is RACF, but other external security manager products can get control through the SAF exit interface, and perform their own **early verification** routine.

Note: During this early invocation of the RACROUTE macro, the user exit ICHRTX00 can be called as part of the SAF processing. If your ICHRTX00 exit is coded in such a way that the ENVIR=VERIFY parameter is ignored, this could cause the SAF to invoke a service that is not valid in problem state.

You should review your use of the SAF user exit ICHRTX00, to make sure that it correctly handles any RACROUTE REQUEST=VERIFY, ENVIR=VERIFY calls in problem state.

CICS defers the creation of the accessor environment element until the RACROUTE REQUEST=VERIFY macro with the ENVIR=CREATE option is issued to perform the **normal verification** routine. The ENVIR=CREATE version of the macro is issued by the security manager domain running in supervisor state.

CICS passes the following information on the ENVIR=VERIFY version of the RACROUTE REQUEST=VERIFY macro:

USERID
The userid of the user signing on to the CICS region.

GROUP
The group name, if specified, of the group into which the user wants to sign on.

PASSWRD
The user's password to verify the userid.

NEWPASS
A new value, if specified, for the user's password. This changes the existing password and is to be used for subsequent signons.

OIDCARD
The contents, if supplied, of an operator identification card.

APPL
The APPLID of the CICS region on which the user is signing on. Which APPLID is passed depends on what is specified as system initialization parameters.

INSTLN
A pointer to a vector of CICS-related information, which you can map using the DFHXSUXP mapping macro. This pointer is valid only if ESMEXITS=INSTLN is specified as a system initialization parameter for the CICS region.

New communications area in INSTLN data: The installation data referenced by the INSTLN parameter is extended by a new field, UXPCOMM. This is a two-word communications area that can be used to pass information between the two phases of the signon verification process—between the early verification routine initiated by ENVIR=VERIFY, and the normal verification routine initiated by ENVIR=CREATE. The 2-word field is defined in the DFHXSUXP DSECT as follows:

```
UXPCOMM  DS    A             Address of 2-word communication area
```

CICS maintains a separate communications area for each task, in CICS-key storage.

Writing an early verification routine

An early verification routine, written for the `ENVIR=VERIFY` option, receives control from SAF in the usual way from the external security manager whose entry point is addressed by field SAFVRACR in the SAF vector table. It receives control in the same state as its caller, as follows:

- Problem-program state
- Task mode (usually the CICS quasi-reentrant TCB)
- PSW storage key 8
- 31-bit addressing mode
- Primary address translation mode.

Register 13 points to a standard 18-word save area. Register 1 points to a 2-word parameter list, where:

- The first word is the address of the SAF parameter list for the VERIFY function,
- The second word is the address of a 152-byte work area.

Using CICS API commands in an early verification routine

An early verification routine can use CICS application programming interface (API) commands, provided it obeys the following interface rules:

- The routine must be written in assembler.
- Entry to the routine must be via the DFHEIENT macro, which saves the caller's registers and establishes a CICS early verification API environment.
- Exit from the routine must be via the DFHEIRET macro, which releases the CICS early verification API environment and restores the caller's registers.
- The routine *must* be link-edited with the special security domain API stub, DFHXSEAI, *instead of* the normal CICS API stub, DFHEAI0. The CICS early verification stub causes linkage to a special interface routine that is aware of the SAF interface linkage requirements, and saves the current CICS command environment. In addition, the standard EXEC interface stub DFHEAI should

also be included, immediately before the early verification routine, with an ORDER statement:

```
INCLUDE SYSLIB(DFHXSEAI)
INCLUDE SYSLIB(DFHEAI)
ORDER   DFHEAI,verify-program,DFHEAI0
ENTRY   verify-program
```

The DFHEIENT and DFHEIRET macros are inserted by the CICS translator unless you specify

```
*ASM XOPTS(NOPROLOG,NOEPILOG)
```

as the first statement of the program. The DFHEIENT macro assumes that register 15 points to its first executable instruction.

Upon return from the DFHEIENT macro, a CICS storage area mapped by the DFHEISTG macro has been established. The pointer DFHEIBP (and the register specified in the EIBREG parameter of DFHEIENT) contains the address of an EXEC interface block (EIB). DFHEICAP contains the pointer to the original parameter list supplied by the SAF interface.

Return and reason codes from the early verification routine

Before returning control, the early verification routine should set a return code and reason code in fields SAFPRRET and SAFPRREA of the SAF parameter list. It should also pass a value to be returned as the SAF return code in a register that is specified in the RCREG keyword of the DFHEIRET macro that is used to exit the program. These return codes are examined by the CICS signon function, and any non-zero value in SAFPRRET is interpreted as a verification failure and causes the signon to fail. A zero return code allows the signon to proceed, and eventually CICS issues a `RACROUTE REQUEST=VERIFY,ENVIR=CREATE` macro in supervisor state and under control of the CICS resource-owning TCB. It is only at this invocation that CICS accepts an ACEE address from the external security manager.

End of Product-sensitive programming interface

Problem determination

There are changes to message and codes, and changes to the trace points for the signon component. For details of the trace entries, see the *CICS/ESA Problem Determination Guide*.

Changes to messages

The following messages are removed:

DFHSN0005	DFHSN0609
DFHSN0109	DFHSN0700
DFHSN0607	DFHSN0701
DFHSN0608	DFHSN0802

The following messages are added:

DFHSN0001	DFHSN1131	DFHSN1300
DFHSN0004	DFHSN1132	DFHSN1410
DFHSN1116	DFHSN1150	DFHSN1501

DFHSN1129	DFHSN1211	DFHSN1850
DFHSN1130	DFHSN1250	DFHSN1851

The following messages are replaced:

DFHSN0100 by DFHSN1100	DFHSN0119 by DFHSN1119
DFHSN0101 by DFHSN1101	DFHSN0120 by DFHSN1120
DFHSN0102 by DFHSN1102	DFHSN0200 by DFHSN1200
DFHSN0103 by DFHSN1103	DFHSN0212 by DFHSN1212
DFHSN0104 by DFHSN1104	DFHSN0213 by DFHSN1213
DFHSN0105 by DFHSN1105	DFHSN0214 by DFHSN1214
DFHSN0106 by DFHSN1106	DFHSN0215 by DFHSN1215
DFHSN0107 by DFHSN1107	DFHSN0400 by DFHSN1400
DFHSN0108 by DFHSN1108	DFHSN0401 by DFHSN1401
DFHSN0112 by DFHSN1112	DFHSN0500 by DFHSN1500
DFHSN0113 by DFHSN1113	DFHSN0604 by DFHSN1604
DFHSN0114 by DFHSN1114	DFHSN0605 by DFHSN1605
DFHSN0115 by DFHSN1115	DFHSN0606 by DFHSN1606
DFHSN0117 by DFHSN1117	DFHSN0800 by DFHSN1800
DFHSN0118 by DFHSN1118	DFHSN0801 by DFHSN1801

Part 6. Problem determination

This Part describes some key changes to problem determination in CICS/ESA 4.1:

Chapter 25. Problem determination changes

This chapter describes changes and additions introduced in CICS/ESA 4.1 which affect CICS problem determination. It covers the following topics:

- New dump exit parameters on the VERBEXIT subcommand
- New information in formatted system dumps
- New component identifiers for trace selectivity
- Application storage protection

New dump exit parameters on the VERBEXIT subcommand

You can selectively format those parts of a system dump that are of interest to you at any particular time, a specific levels of detail, by using the VERBEXIT subcommand in IPCS. You specify, via component keywords, which areas you want the CICS410 dump exit to format dump data for, and the level information specifies the amount of data you want formatted.

With the introduction of new domains at CICS/ESA 4.1, there are new dump component keywords for the directory manager domain (DD), the program manager domain (PG), the security domain (XS), and the user domain (US).

There is also a new TRS component keyword that allows you to exercise selectivity over the formatting and printing of the trace entries written to a system dump. The selectivity provided by the TRS option is the same as that provided for the formatting and printing of trace entries in an auxiliary trace. The trace selectivity parameters that you can specify can be any of the valid trace selectivity parameters available on the trace utility program, DFHTU410.

DD component keyword

The DD component keyword enables you to control the formatting of those parts of a system dump associated with the directory manager domain. The values you can specify on the DD keyword are as follows:

DD[={0|1|2|3}]
The directory manager domain. Specify level 1 to format the directory manager summary; 2 to format directory manager control blocks, including the anchor block, directory headers, and AVL tree headers; 3 to format levels 1 and 2 data; 0 to suppress.

PG component keyword

The PG component keyword enables you to control the formatting of those parts of a system dump associated with the program manager domain. The values you can specify on the PG keyword are as follows:

PG[={0|1|2|3}]
The program manager domain. Specify level 1 to format the program manager summary; 2 for the program manager control blocks, including the anchor block, the LLEs, the PGWEs, the PPTEs, the PLCBs, and the HTBs; 3 to format levels 1 and 2 data; 0 to suppress.

The PCP dump keyword is not applicable at CICS/ESA 4.1.

US component keyword

The US component keyword enables you to control the formatting of user domain control blocks. The values you can specify on the US keyword are as follows:

US[={0|1|2|3}]
The user domain. Specify level 1 to format a summary of users; 2 for user domain control blocks (including the anchor block); 3 to format levels 1 and 2 data; and 0 to suppress.

XS component keyword

The XS component keyword enables you to control the formatting of security manager domain's blocks. The value you can specify on the XS keyword are as follows:

XS[={0|2}]
The security manager domain. Specify level 2 to format the security manager domain anchor blocks; 0 to suppress.

TRS component keyword

The TRS component keyword allows you to exercise much the same selectivity over the formatting and printing of trace entries written to a system dump as you can exercise over the formatting and printing of trace entries in an auxiliary trace.

TRS[={<trace selectivity parameter(s)>}]
Trace entry selectivity.

This keyword is effective only if the TR keyword value is 1, 2, or 3.

The trace selectivity parameter can be any valid trace selectivity parameter available to DFHTU410 for the formatting of CICS auxiliary trace entries except the parameters PAGESZE, ABBREV, and FULL. You can, as with DFHTU410, select any number of parameters from those available.

Note, however, that you must use angled brackets around the parameter, or sequence of parameters, that you specify. The format and default values of parameters used to select trace entries from an internal SDUMP trace, are the same as those that apply when you use DFHTU410P to format auxiliary trace entries.

For more information about using TRS to format trace in a system dump, see “Selecting parts of the CICS internal trace table” on page 558.

New information in formatted system dumps

In CICS/ESA 4.1, there is new information captured in system dumps, which can be formatted using existing VERBEXIT keywords.

Information relating to transaction isolation

In the case of ASRA, ASRB, AICA, and ASRD abends, the kernel error data tells you whether the failing task was running in a subspace or in the base space.

Use the SM keyword to format information relating to the transaction isolation status of tasks in your system. In the SM summary, you are presented with

information regarding storage above and below 16MB, including the number of DU-AL slots used, the number of ALETs stolen.

Summaries of the SDSA, RDSA, and ESDSA are added to the summaries provided in CICS/ESA 3.3 for the CDSA, UDSA, ECDSA, EUDSA, and ERDSA. Each DSA summary also lists the addresses and sizes of the extents (or areas) of storage that it has used.

The system dump also includes a transaction block summary which, essentially, provides storage manager domain's view of the transactions in the system.

New component identifiers for trace selectivity

At CICS/ESA 4.1, there are a number of new components for which you can select special and standard tracing by using the CETR transaction (and the EXEC CICS SET TRACETYPE command). Each new component can be identified with a two–character identifier (which is presented under CETR) as follows:

- DD - directory manager domain
- PG - program manager domain
- US - user domain
- XM - transaction manager domain
- XS - security manager domain.

Application storage protection

Transaction isolation in CICS/ESA 4.1 offers storage protection between application programs, ensuring that one application does not accidentally overwrite the task-lifetime storage of another application program.

Application overwrites can affect the reliability and availability of a CICS system, and the integrity of the data in the system.

Errant programs are now identified as soon as:

- The program attempts to modify storage it does not own, or
- The program passes an address to CICS to which it, the program, does not have write access in order for CICS to modify storage on its, the program's, behalf. In previous releases, CICS does not check ownership of this storage and executes the command with consequent storage violations.

See the Chapter 2, "Transaction isolation" on page 9 for details regarding of the hardware and software requirements as well as the system programming requirements for transaction isolation.

A program attempts to modify storage it does not own

If an application program attempts to modify storage to which it does not have write access, it is abended with abend code ASRA.

It may be that the program attempts to overwrite the code of another application program. In earlier releases, this very probably results in the failure of the overwritten program; a serious problem in a production region. The effect, however, is generally immediate and the program can, in most cases, be recovered, permitting the terminal user to retry the failed program.

If an application program attempts to overwrite the data of another application program, the effect is, very often, not immediate. The overwritten data may lie undetected for a period of time, it may also be written to a database. When the data is next used by an application program, it often causes the program using it to fail or return unpredictable, erroneous results. The end result is often a storage violation. By the time that this actually happens, it is, in most cases, too late to allow the cause of the error to be determined.

In CICS/ESA 4.1, abending tasks that attempt such overwrites greatly decreases the number of storage violations, which, in many cases, are extremely difficult to resolve.

A program attempts to pass an invalid address to CICS

If an application program attempts to pass to CICS an address of storage to which it, the program, does not have access, the transaction is abended with abend code AEYD. This happens when a program issues an API command that causes CICS to modify storage on the program's behalf, yet the program does not own the storage. In earlier releases, CICS does not check ownership of the storage referenced by the passed address, and executes such commands with consequent storage violations, which may be very difficult to resolve.

Chapter 26. CICS dumping in a sysplex

This chapter describes how CICS/ESA 4.1 enables you to capture dump data simultaneously from multiple CICS regions on multiple MVS images in a sysplex. It covers the following topics:

- Overview
- Benefits
- Requirements for multiple address space dumping
- Changes to CICS externals
- Problem determination in a sysplex

Overview

CICS/ESA 4.1 provides simultaneous dump data capture from multiple CICS regions in a sysplex. There are two methods that you can use to gather dump data simultaneously from multiple CICS regions. One requires MVS console operator intervention; the other is initiated automatically.

Automatic dump data capture from related CICS regions

In a sysplex containing many CICS regions connected by MRO links, the resolution of problems may involve using dumps from multiple, related, CICS regions. You may need dump information from CICS regions that reside on more than one MVS image—a task may involve a TOR, an AOR, and an FOR, and possibly additional AORs via DPL, and some of these may be running on different MVS images.

It is not always immediately apparent, in such a complex environment where many regions are interconnected, which regions are doing related work and, therefore, from which regions you need dump data in the event of an error. Indeed, by the time that an error has developed on one system, and the dump has been taken, processing may have continued on related regions with the strong possibility that data which may have been useful in the resolution of the original problem has been overwritten and lost for diagnosis purposes.

A related CICS region is one in which the unit of work identifiers, in the form of APPC tokens, of one or more tasks match those in the CICS region that issued the dump request.

This facility allows the simultaneous capture of dump data on all related CICS regions.

You can control this dumping by means of the DUMPSCOPE option on each CICS dump table entry. This option dictates whether dumps of all related CICS regions are taken when a dump with the dump code defined as having DUMPSCOPE=RELATED is taken.

The facility exploits the services of XCF/MRO and the MVS workload manager to send requests for SDUMPs to all MVS images in a sysplex that are running related CICS regions.

Operator-requested simultaneous dump data capture

An MVS operator may want to issue a dump request to several CICS regions. Perhaps a number of CICS regions are hung - there is no way of telling where the problem causing the hang might lie. The operator needs to be able to issue a request for system dumps simultaneously to all regions in the sysplex. Until CICS/ESA 4.1, this has not been possible.

The MVS operator can exploit MVS support for remote SDUMPs. This is available with MVS/ESA 5.1. for MVS images that are XCF-connected.

Benefits

This facility reduces operating costs involved in the diagnosis of CICS system problems.

Benefits of automatic simultaneous dumps

Where a task involves multiple CICS regions, and a dump is taken on one system, usually in response to an error situation, it is very often the case that dump data from all related CICS regions is needed to fully diagnose and solve the error.

It is difficult to identify which regions are related in some complex environments, and, therefore, it is difficult to gather all the dump data required to resolve error situations.

This facility allows you to capture, automatically, dump data of all related CICS regions in a sysplex.

This means that the required documentation is collected at first failure, reducing operating costs involved in the diagnosis of CICS system problems.

Benefits of operator-requested simultaneous dumps

As already described, this facility is beneficial where there has been a dump taken in one CICS region and dumps are needed in all related CICS regions. This facility is also particularly beneficial in situations where one or more CICS regions in a sysplex environment are hung, and an MVS console operator needs to issue a dump request to other, related, CICS regions in the sysplex that might be responsible for causing the hang.

Without this facility, you cannot request dumps from all these regions to be taken at exactly the same time, with the resultant problems for diagnosis and problem determination.

With this facility, the MVS console operator, facing a situation in which some CICS regions are hung, has only to issue one dump request to ensure that all required dump data, from all related CICS regions on all MVS images in the sysplex, is gathered.

Requirements for simultaneous dumps

For this facility you require:

- The software required to create a sysplex that supports XCF/MRO. This is:
  - MVS/ESA 5.1
  - The CICS/ESA 4.1 interregion communication program, DFHIRP
  - MVS workload manager.

  The hardware to support a sysplex:
  - Channel-to-channel links, ESCON channels, or high-speed coupling facility links
  - External time reference facility
  - XCF coupled data sets.

You must ensure that the CICS regions that are to benefit from this facility communicate using CICS interregion communication (IRC). This causes the CICS regions to join the XCF group, DFHIR000. CICS interregion communication must be started in each of the related CICS regions, using either the ISC and IRCSTRT system initialization parameters, or a SET IRC OPEN command.

Changes to CICS externals

The introduction of this facility results in a number of changes to CICS externals. These are:

- Changes to the system programming interface
- Changes to CICS-supplied transactions.

Changes to the system programming interface

The CICS system programming interface (SPI) is extended with changes to the following commands:

- EXEC CICS INQUIRE SYSDUMPCODE
- EXEC CICS SET SYSDUMPCODE
- EXEC CICS INQUIRE TRANDUMPCODE
- EXEC CICS SET TRANDUMPCODE

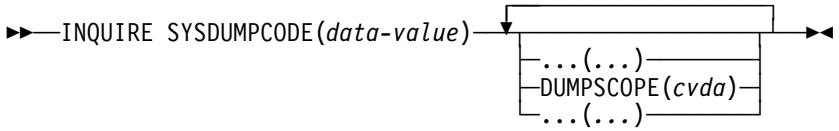
EXEC CICS INQUIRE SYSDUMPCODE

General-use programming interface

Function

Retrieve information about system dump codes.

Syntax



Conditions:
END, ILLOGIC, NOTAUTH, NOTFND

INQUIRE SYSDUMPCODE options

DUMPSCOPE

returns a CVDA value identifying whether SDUMP requests of dumps with the specified dump code are to be sent to MVS images in the sysplex that are running XCF/MRO connected CICS regions that are related to the CICS region initiating the dump request. CVDA values are:

LOCAL

SDUMP requests are not to be sent to MVS images in the sysplex that are running XCF/MRO connected CICS regions.

RELATED

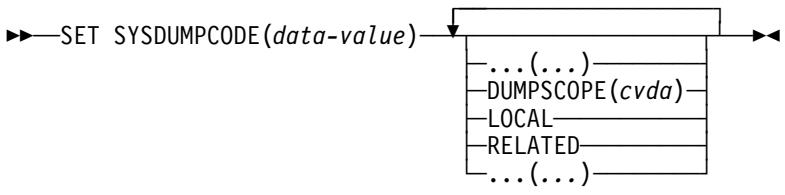
When an SDUMP is requested for the specified dump code, the request is sent to all MVS images in the sysplex that are running XCF/MRO connected CICS regions, which are related to the CICS region initiating the dump. The CICS regions must be executing with MVS/ESA 5.1 and the MVS workload manager.

EXEC CICS SET SYSDUMPCODE

Function

Change entries in the system dump code table.

Syntax



Conditions:
DUPREC, INVREQ, IOERR, NOSPACE, NOTAUTH, NOTFND

SET SYSDUMPCODE options

DUMPSCOPE

specifies whether SDUMP requests for dumps with the specified dump code are to be sent to all other MVS images in the sysplex that are running XCF/MRO connected CICS regions, which are related to the CICS region that initiated the dump request. CVDA values are:

LOCAL

SDUMP requests are not to be sent to MVS images in the sysplex that are running XCF/MRO connected CICS regions. A dump of the local CICS region only is to be taken. This is the default.

RELATED

When an SDUMP is requested for the dump code, this request is sent to all MVS images in the sysplex that are running XCF/MRO connected CICS regions, which are related to the CICS region that initiated the dump request. A dump of each of these CICS regions containing related work is then taken, and the dumps are written to the MVS SYS1.DUMPxx data sets on their MVS image. The CICS regions must be executing with MVS/ESA 5.1 and the MVS workload manager.

SET SYSDUMPCODE conditions

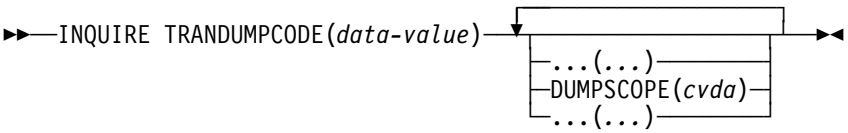
Condition	*RESP2*	*Meaning*
INVREQ	13	DUMPSCOPE has an invalid CVDA value
INVREQ	14	DUMPSCOPE=RELATED requires MVS/ESA 5.1

EXEC CICS INQUIRE TRANDUMPCODE

Function

Retrieve information about transaction dump codes.

Syntax



Conditions:
END, ILLOGIC, NOTAUTH, NOTFND

INQUIRE TRANDUMPCODE options

DUMPSCOPE

returns a CVDA value identifying whether SDUMP requests of dumps with the specified dump code are to be sent to MVS images in the sysplex that are running XCF/MRO connected CICS regions that are related to the CICS region initiating the dump request. CVDA values are:

LOCAL

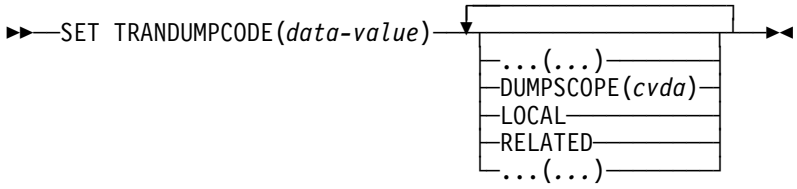
SDUMP requests are not to be sent to MVS images in the sysplex that are running XCF/MRO connected CICS regions.

RELATED
When an SDUMP is requested for the specified dump code, the request is sent to all MVS images in the sysplex that are running XCF/MRO connected CICS regions, which are related to the CICS region initiating the dump. The CICS regions must be executing with MVS/ESA 5.1 and the MVS workload manager.

EXEC CICS SET TRANDUMPCODE

Function
Change entries in the transaction dump code table.

Syntax



Conditions:
DUPREC, INVREQ, IOERR, NOSPACE, NOTAUTH, NOTFND

SET TRANDUMPCODE options
DUMPSCOPE
specifies whether SDUMP requests for dumps with the specified dump code are to be sent to all other MVS images in the sysplex that are running XCF/MRO connected CICS regions, which are related to the CICS region that initiated the dump request. CVDA values are:

LOCAL
SDUMP requests are not to be sent to MVS images in the sysplex that are running XCF/MRO connected CICS regions. A dump of the local CICS region only is to be taken. This is the default.

RELATED
When an SDUMP is requested for the dump code, this request is sent to all MVS images in the sysplex that are running XCF/MRO connected CICS regions, which are related to the CICS region that initiated the dump request. A dump of each of these CICS regions containing related work is then taken, and the dumps are written to the MVS SYS1.DUMPxx data sets on their MVS image. The CICS regions must be executing with MVS/ESA 5.1 and the MVS workload manager.

SET TRANDUMPCODE conditions

Condition	*RESP2*	*Meaning*
INVREQ	13	DUMPSCOPE has an invalid CVDA value
INVREQ	14	DUMPSCOPE=RELATED requires MVS/ESA 5.1

End of General-use programming interface

Changes to CICS-supplied transactions

The CEMT command is modified to allow you to inquire upon and set the RELATED and LOCAL option with for system dump codes and transaction dump codes. The following commands have the added function:

- CEMT INQUIRE SYDUMPCODE
- CEMT SET SYDUMPCODE
- CEMT INQUIRE TRDUMPCODE
- CEMT SET TRDUMPCODE

CEMT INQUIRE SYDUMPCODE

Function

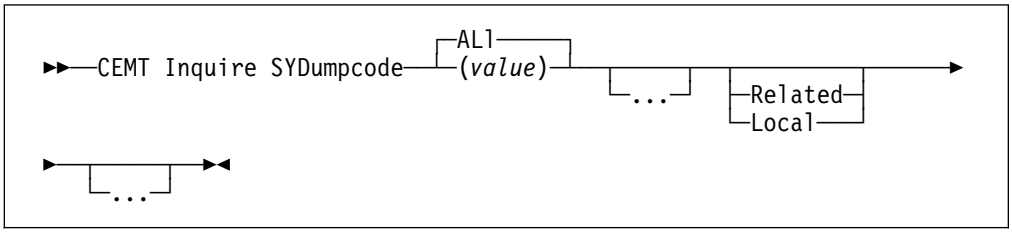
Retrieve information about the system dump code table.

Context

INQUIRE SYDUMPCODE allows you to see the current settings of the entries in the system dump table.

The INQUIRE command tells you whether a system dump request is to be sent to other MVS images in the sysplex that are running CICS regions connected via XCF/MRO to the system on which this command is issued.

Syntax



INQUIRE SYDUMPCODE options

Related**|L**ocal

displays whether a system dump request is to be sent to MVS images in the sysplex that are running XCF/MRO connected CICS regions related to the CICS region on which the dump is initiated.

Related A system dump request is to be sent for this system dump code.

Local A system dump request is not to be sent to other MVS images in the sysplex for this system dump code.

CEMT SET SYDUMPCODE

Function

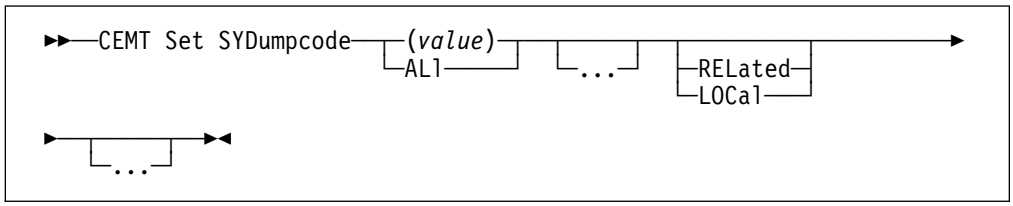
Change the attributes of the system dump codes.

Context

SET SYDUMPCODE allows you to define the entries that you require in the system dump table. You control system dumps by creating an entry in the system dump table for each dump code that requires a change from the default action.

The SET SYDUMPCODE allows you to specify whether a system dump request is to be sent to all MVS images in the sysplex that are running XCF/MRO connected CICS regions related to the region that initiated the dump request. A system dump is then taken for each of these related regions.

Syntax



SET SYDUMPCODE options

RELated|LOCal

specifies whether a system dump request is to be sent to all MVS images in the sysplex that are running XCF/MRO connected CICS regions related to the CICS region that initiated the dump request.

RELATED

a system dump request is to be sent for this system dump code.

LOCAL

a system dump request is not to be sent to other MVS images in the sysplex for this system dump code. This is the default.

CEMT INQUIRE TRDUMPCODE

Function

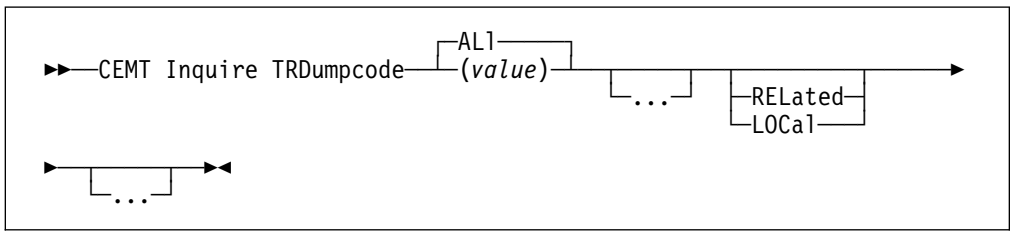
Retrieve information about transaction dump codes.

Context

The INQUIRE function allows you to see the current settings of the entries in the transaction dump table.

The INQUIRE command tells you whether a system dump request is to be sent to MVS images in the sysplex that are running XCF/MRO connected CICS regions related to the CICS region that initiated the dump request. These related CICS regions then take a system dump.

Syntax



INQUIRE TRDUMPCODE options

Related**|L**ocal

displays whether a system dump request is to be sent to MVS images in the sysplex that are running XCF/MRO connected CICS regions related to the region which initiated the dump request. A system dump is then taken for each of these related regions. The values are:

Related

A system dump request is to be sent for this transaction dump code.

Local

A system dump request is not to be sent to other MVS images in the sysplex.

CEMT SET TRDUMPCODE

Function

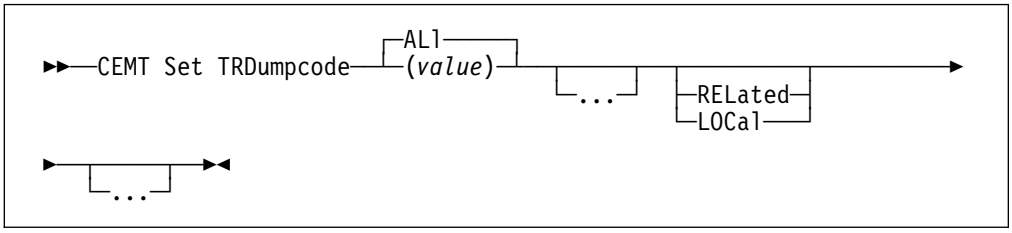
Change the attributes of the transaction dump codes.

Context

The INQUIRE function allows you to see the current settings of the entries in the transaction dump table.

The SET TRDUMPCODE allows you to specify whether a system dump request is to be sent to all MVS images in the sysplex that are running XCF/MRO connected CICS regions related to the CICS regions which initiated the dump request. A system dump is then taken for each of these related regions.

Syntax



SET TRDUMPCODE options

Related**|L**ocal

specifies whether a system dump request is to be sent to MVS images in the sysplex that are running XCF/MRO connected CICS regions related to the CICS region which initiated the dump request. A system dump is then taken for each of these related regions. The values are:

Related

A system dump request is to be sent for this transaction dump code.

Local

A system dump request is not to be sent to other MVS images in the sysplex. This is the default.

Problem determination

This facility considerably aids problem determination in XCF/MRO environments where many CICS regions are running. Two types of situation benefit from this:

- Where a task involves multiple CICS regions in a sysplex, and one region issues a dump, typically in response to an error.

  Dump data from all CICS regions related to the region issuing the dump request is usually required in order to fully diagnose and solve the problem. This dump data from related regions would also need to be captured at the same time as the dump taken on the region issuing the dump.

- Where an MVS console operator needs to capture, simultaneously, dump data from multiple CICS regions in the sysplex.

Before CICS/ESA 4.1, such automatic and simultaneous dump data capture of CICS data was impossible.

You need MVS/ESA 5.1, the MVS workload manager, and the XCF facility in order to use this. The MVS images in the sysplex must be connected via XCF. The CICS regions must be using MRO supported by the CICS/ESA 4.1 interregion communication program, DFHIRP.

Automatic dump data capture from related CICS regions

It is possible to collect dump data simultaneously from all related CICS regions in a sysplex. Related CICS regions are those containing one or more tasks which have unit of work identifiers, in the form of LU6.2 tokens, that match the unit of work identifiers in the CICS region which initially issued the dump request.

The CICS regions must be connected via XCF/MRO. Connections using VTAM ISC are not eligible to use the related dump facility.

The function is controlled by the DUMPSCOPE option on each CICS dump table entry. You can set this option to have either of the following values:

- RELATED - take dumps for all related CICS regions across the sysplex.
- LOCAL - take dumps for the requesting CICS region only. This is the default.

The DUMPSCOPE option is available on the following master terminal and system programming commands:

- EXEC CICS INQUIRE SYSDUMPCODE
- EXEC CICS SET SYSDUMPCODE
- EXEC CICS INQUIRE TRANDUMPCODE
- EXEC CICS SET TRANDUMPCODE
- CEMT INQUIRE SYDUMPCODE
- CEMT SET SYDUMPCODE
- CEMT INQUIRE TRDUMPCODE
- CEMT SET TRDUMPCODE

If the DUMPSCOPE option is set to RELATED in the CICS region issuing the dump request, a request for a system dump is sent to all MVS images in the sysplex that run related CICS regions.

The local MVS image running the CICS region that initiated the dump request has two dumps - one of the originating CICS region, the other containing the originating

CICS region and up to fourteen additional related CICS regions from the local MVS image. Any other MVS image in the sysplex that is running related CICS regions has one dump that contains all the related CICS regions in that MVS image.

When a dump is requested, the DUMPSCOPE option is tested only in the CICS region issuing the original dump request. If the requesting CICS region has DUMPSCOPE defined as RELATED for the dump code, then all related CICS regions are dumped even if they have DUMPSCOPE defined as LOCAL for the dump code.

There is a maximum of fifteen related address spaces per MVS image in an SDUMP. If there are more than fifteen related CICS regions on an MVS image, then not all of them will be dumped, the dumps being taken in address space id (ASID) order. Related CICS regions may also fail to be dumped if they are swapped out when the dump request is issued. You should consider whether to make certain CICS regions non-swappable as a result.

Operator-requested simultaneous dump data capture

The MVS console operator may want to issue a dump request simultaneously to several CICS regions in the sysplex. There may be a problem in the sysplex where one or more CICS regions is hanging and, to fully diagnose and solve the problem, dump data captured at the same time is required.

Without this facility, such simultaneous dump data capture across multiple CICS regions in the sysplex is impossible.

Use the following command at the console:

```
DUMP COMM=()
R x,REMOTE=(SYSLIST=*),PROBDESC=(SYSDCOND,SYSDLOCL,(DFHJOBN,jobnames))
```

where:

- **REMOTE** controls the issuing of dumps on remote systems.
- **SYSLIST=*** means the request is to be routed to all remote systems.
- **PROBDESC** is problem description information, as follows:
  - SYSDCOND - an MVS keyword. This specifies that a dump is to be taken on remote MVS images if the IEASDUMP.QUERY exit responds with return code 0. CICS supplies DFHDUMPX as the IEASDUMP.QUERY exit.
  - SYSDLOCL - an MVS keyword. This drives the IEASDUMP.QUERY exit on the local and remote MVS images. This allows the CICS regions on the local MVS region to be dumped.
  - DFHJOBN - a CICS keyword. The operator should include the generic job name. This is used by DFHDUMPX to determine which address spaces to dump.

If you adopt a suitable naming convention for your CICS regions, this can be used to define suitable generic jobnames to determine which CICS regions to dump. See the *System/390 MVS Sysplex Application Migration* manual for recommendations on naming conventions. If you follow the recommendation in this manual, the generic job name for all CICS regions in the sysplex would be 'CICS*'.

Part 7. General enhancements

This Part describes a number of general changes and enhancements, many of which are introduced to meet outstanding user requirements. It includes the following topics:

Chapter 27. Changes to database management

This chapter, describes the changes to database management in CICS/ESA 4.1. It contains the following topics:

- CICS DB2 attachment facility
- DBCTL operator transaction, CDBM
- Specifying a DBCTL system identifier (DBCTLID)
- Release DBCTL threads at syncpoint
- Issue IMS AIB call format
- Removal of support for IMS/VS 2.2 with local DL/I
- Resource manager interface global user exits, XRMIIN and XRMIOUT
- DBCTL installation verification procedure.

Note: CICS/ESA 4.1 is expected to be the last release of CICS/ESA to support the local DL/I interface to IMS/ESA, leaving DBCTL as the only supported interface to IMS databases. CICS/ESA 4.1 provides a number of enhancements to the CICS-DBCTL interface, notably an installation verification procedure (IVP) to ease migration to DBCTL, and a CICS-supplied transaction, CDBM, to simplify operator communication between CICS and DBCTL.

CICS DB2 attachment facility

The CICS DB2 attachment facility is shipped on the CICS/ESA 4.1 product tape with a number of enhancements. The CICS DB2 attachment facility works with all supported releases of DB2.

You must use the CICS DB2 attachment facility to connect a CICS/ESA 4.1 region to DB2. The DB2 program product continues to supply the earlier version of the attachment facility to support DB2 connections with releases of CICS earlier than CICS/ESA 4.1.

The CICS DB2 attachment facility includes the following enhancements:

- The attachment facility runs above 16MB.
- The resource control table has a two-byte suffix.
- The DSNCRCT*x* PARM parameter is replaced by the INITPARM system initialization parameter, and the default RCT suffix is 00.

  The syntax of the INITPARM system initialization parameter for DB2 is as follows:

  INITPARM=(DSN2STRT='*rct_suffix,DB2_subystem_id*','other_initparms')
- You can specify the RCT suffix and DB2 subsystem ID on the CICS system initialization parameter INITPARM, and also on the DSNC STRT command.
- SQL programs can run in PLT initialization after DSN2COM0 has run.
- Attachment facility traces provide more information.
- The new attachment facility has some performance benefits over old attachment facility.
- You can now identify waits for DB2 on CEMT I TAS and on DB2 formatted region dumps.

- The resource control table provides three new options:
  - TXIDSO, to allow suppression of some signons during thread reuse
  - PLANI on TYPE=INIT to specify a default plan name
  - PURGEC, to specify the lebgth of the protected thread cycle purge.
- A recovery restriction has been removed
- You can use the INQUIRE EXITPROGRAM command to determine whether the attachment facility is started, and thus prevent AEY9 abends.

Benefits

The enhanced CICS DB2 attachment facility offers a number of benefits compared with the attachment facility shipped with DB2.

Virtual storage constraint relief

The CICS DB2 attachment facility modules and control block storage reside above 16MB, providing virtual storage constraint relief and improved performance.

The attachment facility runs above 16MB, and is enabled using the LINKEDITMODE option, which allows you to specify TASKDATLOC(ANY) for your application programs that use SQL. (See the *CICS/ESA System Programming Reference* for information about the LINKEDITMODE option of the EXEC CICS ENABLE command.)

Greater flexibility and operation of the RCT

A two-byte resource control table suffix enables you to have many more versions of the resource control table, and to have greater flexibility with your naming conventions.

The ability to select the resource control table, and the DB2 subsystem ID, using either the CICS system initialization parameter INITPARM or the DSNC STRT command. This means you can use a common RCT for different DB2 subsystems and override the subsystem ID using one of these two methods.

Improved tracing

Tracing is done using CICS system trace, which means that additional information can be returned and there is no need for user tracing. You control tracing in the CICS DB2 attachment facility using the CICS file control trace options, setting level 1 or 2 as required.

The CICS system trace also gives an improvement in performance compared with the previous method of tracing in the attachment facility.

Performance improvements

Pathlength reductions designed to improve performance are achieved by a redesigned timer mechanism for purging threads, and by replacing EXEC CICS WAIT EXTERNAL commands by XPI WAIT_MVS calls.

The latter change also enables CEMT INQUIRE TASK commands and CICS formatted region dumps to indicate tasks that are waiting on DB2.

Status information improvement

The EXEC CICS INQUIRE EXITPROGRAM(DSN2EXT1) CONNECTST command enables you to determine the connection status of CICS with its DB2 subsystem.

You can use the information returned to ensure that your application programs do not issue SQL requests when DB2 is not available, and thereby avoid AEY9 abends.

Changes to CICS externals

The CICS/ESA 4.1 CICS DB2 attachment facility causes the following changes to CICS externals:

- Changes to installation of resource control tables
- Changes to system definition
- Changes to resource definition
- Changes to the system programming interface.

Changes to installation of resource control tables

DSN2CTxx is the resource control table for the CICS DB2 attachment facility, replacing DSNCRCT.

You can use the DFHAUPLE procedure to assemble and link edit your resource control table. The following is an example of JCL that you can use to assemble your resource control table:

```
//RCT      EXEC DFHAUPLE,
//         PARM.LNKEDT='AMODE=24,RMODE=24,LIST,XREF,LET,NCAL'
//ASSEM.SYSUT1 DD DSN=DSNxxx.SDSNSAMP(DSN8FRCT),DISP=SHR
//LNKEDT.SYSLMOD DD DSN=DSNxxx.SDSNLOAD,DISP=SHR
```

This job does not need to reference the DB2 macro library (DSNxxx.SDSNMACS). However, if the DB2 macro library has been added to the SYSLIB statement of the ASM step in the DFHAUPLE procedure, it must be concatenated *after* the CICS macro library. The LNKEDT.SYSLMOD library can be any library that is concatenated in the STEPLIB DD statement of your CICS startup JCL.

Note: The resource control table must be linked RMODE(24).

Your CICS startup JCL must observe the following:

- DSNxxx.SDSNLOAD library must be concatenated *after* the CICS libraries in the STEPLIB DD statement
- The library containing the DB2 resource control table must also be included on the STEPLIB DD statement.

APAR PN88915

Documentation for PN88915 added on 24 October 1996

Note: Generally, you do not need any DB2 libraries in the DFHRPL DD statement. If you do need DB2 libraries in the DFHRPL concatenation for an application, they should be placed after the CICS libraries. For example, you need SDSNLOAD in the DFHRPL to support those applications that issue dynamic calls to the DB2 message handling module, DSNTIAR.

You do not need to use the DSNTIJSU installation job to link-edit the stubs DFHEAI and DFHEAI0 because the CICS DB2 attachment facility modules are supplied with the stubs pre-linked.

CICS/ESA 4.1 supplies a CSD group, DFHDB2, as part of the standard DFHLIST list. DFHDB2 contains the CSD definitions for the CICS DB2 attachment facility programs and transactions. These include different values for the TASKDATALOC and DATALOCATION parameters for attached modules to enable the CICS DB2 attachment facility to run in 31-bit addressing mode. These modules, which run above 16MB, are installed automatically as part of the CICS/ESA 4.1 installation process.

Renaming DB2 modules in PLTs: If you have program DSNCCOM0 in your startup PLT, you must rename it as DSN2COM0. If you have program DSNCCOM2 in your shutdown PLT, you must rename it as DSN2COM2.

Changes to system definition

This section describes:

- Connecting DB2 to CICS automatically
- Running SQL programs during PLT processing
- Additional options on the resource control table

Connecting DB2 to CICS automatically: In CICS/ESA 4.1, you can specify the resource control table suffix and the DB2 subsystem ID when connecting CICS to DB2.

You can do this using the DB2 command:

```
DSNC STRT xx,yyyy
```

where *xx* is the 2-character resource control table suffix and *yyyy* is the 4-character DB2 subsystem ID.

Alternatively, you can specify an entry for DSN2STRT in the PLTPI, using the INITPARM parameter to pass the RCT suffix and ID to DSN2STRT, as follows:

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

where *xx* is the resource control table suffix and *yyyy* is the 4-character DB2 subsystem ID.

If you specify a DB2 subsystem ID in INITPARM, it is used during PLT processing at startup, or by a DSNC command that omits to specify a resource control table suffix and DB2 subsystem ID.

Table 36 on page 445 gives examples of how the values specified determine which DB2 suffix and subsystem ID are used.

Table 36. Selection of DB2 suffix and subsystem ID			
INITPARM parameter	**DSNC STRT command**	**DB2 suffix**	**DB2 subsystem ID**
INITPARM=AA,BBBB	None	AA	BBBB
INITPARM=AA,BBBB	DSNC STRT CC	CC	BBBB
INITPARM=AA,BBBB	DSNC STRT ,DDDD	AA	DDDD
INITPARM=AA,BBBB	DSNC STRT EE,FFFF	EE	FFFF
None specified	None specified	00 (default suffix)	Taken from resource control table

Running SQL programs during PLT processing: CICS/ESA 4.1 enables you to run SQL programs during PLT processing. In previous releases, CICS does not complete connection to DB2 during PLT processing, so that running an SQL program during PLT, or immediately after, results in an AEY9 abend.

Changes to resource definition

As the CICS DB2 attachment facility runs above 16MB, and because the CICS DB2 attachment facility is enabled with the LINKEDITMODE option, you can specify TASKDATALOC(ANY) so that your programs can run in 31-bit addressing mode.

Changes to CICS trace

You do not need to set both the master trace and user trace flags on to get CICS DB2 attachment facility traces. Instead, you need only set on the system master trace flag, using either the SYSTR system initialization parameter or the CETR transaction. (The trace flag used is the same as for DBCTL.)

See “Trace points in DSN2CTxx” on page 447 and “Trace information returned for the CICS DB2 attachment facility” on page 447 for information on trace IDs in DSN2CTxx, details of information returned, and examples of trace entries.

Changes to the system programming interface

You can use the CONNECTST option of the EXEC CICS INQUIRE EXITPROGRAM(DSN2EXT1) command to determine if CICS is connected to a DB2 subsystem. If the value returned is CONNECTED, CICS is connected to DB2 and application programs can issue SQL calls.

If the value returned is NOTCONNECTED, or if the PGMIDERR condition is returned, DB2 is not available and calls to DB2 will result in an AEY9 abend. See page 219 for full details of this new SPI command.

Note that the INQUIRE EXITPROGRAM command also has a STARTSTATUS option, which enables you to determine whether an exit is enabled, and if so whether it is started or stopped. However, for DB2 you should use the CONNECTST option to determine whether the CICS DB2 attachment facility is started.

The following example illustrates the use of the CONNECTST option on the INQUIRE EXITPROGRAM to determine whether the CICS DB2 attachment facility is started or stopped.

```
DCL RESPONSE  BINARY(31);                    /* RESPONSES TO CICS CMDS */

DCL EXITNAME  CHAR(8) CONSTANT('DSN2EXT1'); /* NAME OF DB2 TRUE       */

DCL ENTRYNAME CHAR(8) CONSTANT('DSNCSQL');  /* ENTRY POINT OF TRUE    */

DCL CONN_STATUS FIXED(31);                   /* CVDA FOR TRUE STATUS   */

⋮
   EXEC CICS INQUIRE EXITPROGRAM(EXITNAME)  ENTRYNAME(ENTRYNAME)
                     CONNECTST(CONN_STATUS) NOHANDLE;

   IF CONN_STATUS = DFHVALUE(CONNECTED)
      THEN ...                           /* DB2 adaptor is started */
```

Recovery

Use of the CICS recovery qualifier corrects a restriction affecting recovery of multiple DB2 subsystems in earlier releases of CICS. The problem resolved by the CICS DB2 attachment facility in CICS/ESA 4.1 is described in the following scenario:

- CICS connects to DB2A
- DB2A terminates abnormally, leaving an in-doubt
- CICS connects to DB2B
- The CICS DB2 attachment facility informs CICS that DB2 has no in-doubts to process
- CICS does not distinguish between different DB2 subsystems, and writes a “forget” record for the in-doubt on DB2A
- When CICS later connects to DB2A, and DB2A sends the in-doubt to CICS, CICS replies that it should not be in-doubt.

In CICS/ESA 4.1, the CICS DB2 attachment facility adds the DB2 subsystem ID to the CICS recovery qualifier to distinguish one DB2 from another. During recovery, CICS only deals with the in-doubts for the specified DB2, and so it can remember in-doubts across different DB2 connections.

Performance

All CICS DB2 attachment facility modules and control block storage are moved above 16MB, and DB2 programs can run in 31-bit addressing mode. This removes the need for CICS to copy any storage from above 16MB to below, thus easing any constraints on virtual storage.

Storage requirements

The CICS DB2 attachment facility virtual storage requirements are about the same as in previous releases, but all virtual storage is moved above 16MB.

Problem determination

This section describes:

- Trace points in DSN2CTxx
- Trace information returned for the CICS DB2 attachment facility
- Examples of trace information for CICS DB2 attachment facility
- Changed DB2 message number prefix
- What to do when the CICS DB2 attachment facility fails to initialize.

Trace points in DSN2CTxx

You can specify a greater range of trace IDs in DSN2CTxx; valid numbers are between 256 and 511 (X'100' through X'1FF'). The **default** trace IDs are 448-450 (X'C0' through X'1C0'). In earlier releases, the default trace IDs were 192-194 (X'C0' through X'C2').

Trace information returned for the CICS DB2 attachment facility

The following trace information is returned for the CICS/ESA 3.3 and earlier CICS DB2 attachment facility:

- Parm 2: History flags, return code
- Parm 3: Plan name

  The following additional information is returned in CICS/ESA 4.1:

- Parm 1: Eyecatcher
- Parm 4: CLOT address
- Parm 5: CCCT address — not available on dynamic plan entries
- Parm 6: CSUB address — not available on dynamic plan entries
- Parm 7: The CLOT, up to the save area fields. Parm 7 is only returned if FC level 2 is specified, or if an EXCEPTION is received, as shown in Figure 27 on page 448.

Examples of trace information for CICS DB2 attachment facility

This section shows a number of examples of trace information produced for the CICS DB2 attachment facility.

SQL call with FC level 1 tracing active: Figure 26 shows an SQL call with FV level 1 tracing active.

```
AP 01C0 USER  EVENT - USER-EXIT-PROGRAM-ENTRY

             TASK-00366 KE_NUM-000F TCB-007DBE88 RET-832053EA TIME-16 14 41 1987713447 INTERVAL-**.**********    =000001=
               1-0000  C4C2F240 6040C5E5 C5D5E3                                        *DB2 - EVENT                  *
               2-0000  01010104 00000000                                              *........                     *
               3-0000  E3C5E2E3 C3F0F540                                              *TESTC05                      *
               4-0000  0311B09C                                                       *....                         *
               5-0000  04008E08                                                       *....                         *
               6-0000  040020D0                                                       *....                         *
```

Figure 26. SQL call with FC level 1 tracing active

SQL call with FC level 2 tracing active: Figure 27 on page 448 shows an SQL call with FC level 2 tracing active.

```
AP 01C0 USER  EVENT - USER-EXIT-PROGRAM-ENTRY

             TASK-00412 KE_NUM-000C TCB-007DBE88 RET-832053EA TIME-16 28 45 0051020634 INTERVAL-01.7423419997*   =000002=
               1-0000  C4C2F240 6040C5E5 C5D5E3                                        *DB2 - EVENT                    *
               2-0000  01010104 00000000                                               *........                       *
               3-0000  E3C5E2E3 C3F0F540                                               *TESTC05                        *
               4-0000  0311B09C                                                        *....                           *
               5-0000  04008E04                                                        *....                           *
               6-0000  040020D0                                                        *....                           *
               7-0000  C4E2D5F2 D3D6E340 000B0000 0001CD98  04008E04 040020D0 E2E8E2C1 C4D44040  *DSN2LOT .......q........SYSADM  *
                 0020  40404040 40404040 E7C3F0F5 00000000  00000000 00000000 00000000 C4C2F2D5  *        XC05................DB2N*
                 0040  C5E34040 E3F1F3F0 F2404040 17D25624  D1B60000 01010104 00000000 00000000  *ET  T1302   .K..J................*
                 0060  00000000 00000000 01010104 00000000  02F00368 00000000 00000000 00000000  *.................0..............*
                 0080  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                 00A0  E3C5E2E3 C3F0F540 01C00000 00000000  00404040 40404040 40070484 00000000  *TESTC05 .........        ..d....*
                 00C0  00000000 0000                                                     *......                         *
```

Figure 27. SQL call with FC level 2 tracing active

Example of DYNAMIC PLAN selection: This section shows an example of DYNAMIC PLAN selection. Figure 28 is an 1C1, dynamic plan entry, with no plan name.

```
AP 01C1 USER  EVENT - USER-EXIT-PROGRAM-ENTRY

             TASK-00428 KE_NUM-000E TCB-007DBE88 RET-832053EA TIME-16 33 54 1079612197 INTERVAL-**.**********    =000001=
               1-0000  C4C2F240 6040C4E8 D5C1D4C9 C340D7D3  C1D540C5 D5E3D9E8              *DB2 - DYNAMIC PLAN ENTRY       *
               2-0000  5C5C5C5C 00000000                                               *****....                       *
               3-0000  00000000 00000000                                               *........                       *
               4-0000  0311B09C                                                        *....                           *
```

Figure 28. DYNAMIC PLAN selection — 1C2 dynamic plan entry

Figure 29 is an 1C2, dynamic plan exit, including a plan name.

```
AP 01C2 USER  EVENT - USER-EXIT-PROGRAM-ENTRY

             TASK-00428 KE_NUM-000E TCB-007DBE88 RET-832053EA TIME-16 33 54 1080917822 INTERVAL-00.0001305625    =000002=
               1-0000  C4C2F240 6040C4E8 D5C1D4C9 C340D7D3  C1D540C5 E7C9E340              *DB2 - DYNAMIC PLAN EXIT        *
               2-0000  5C5C5C5C 00000000                                               *****....                       *
               3-0000  E3C5E2E3 C3F0F540                                               *TESTC05                        *
               4-0000  0311B09C                                                        *....                           *
```

Figure 29. DYNAMIC PLAN selection — 1C2 dynamic plan exit

Exception trace entries: For an exception trace entry, two trace entries are produced, as in earlier versions of the CICS DB2 attachment facility. Figure 30 on page 449 shows an example of the exception trace generated when the number of threads is insufficient.

```
AP 01C0 USER  EVENT - USER-EXIT-PROGRAM-ENTRY

              TASK-00045 KE_NUM-0016 TCB-007DBD18 RET-832013EA TIME-09 11 32 8566297509 INTERVAL-**.**********    =000001=
                1-0000  C4C2F240 60405CC5 E7C35C                                          *DB2 - *EXC*                    *
                2-0000  01000004 00000000                                                 *........                       *
                3-0000  E3C5E2E3 C3F0F540                                                 *TESTC05                        *
                4-0000  0311F46C                                                          *..4%                           *
                5-0000  00000000                                                          *....                           *
                6-0000  00000000                                                          *....                           *
                7-0000  C4E2D5F2 D3D6E340 000B0000 0001CD98  00000000 00000000 E2E8E2C1 C4D44040  *DSN2LOT .......q........SYSADM  *
                  0020  40404040 40404040 E7C3F0F5 00000000  00000000 00000000 00000000 C4C2F2D5  *        XC05................DB2N*
                  0040  C5E34040 E3F1F3F0 F3404040 18B370D1  3B050000 01000004 00000410 00000000  *ET  T1303   ...J.................*
                  0060  00000000 00000000 01000004 00000000  02F10368 00000000 00000000 00000000  *.................1..............*
                  0080  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                  00A0  E3C5E2E3 C3F0F540 01C00000 00000000  00000000 00000000 00000000 00000000  *TESTC05 ........................*
                  00C0  00000000 0000                                                     *......                         *

AP 01C0 USER  EVENT - USER-EXIT-PROGRAM-ENTRY

              TASK-00045 KE_NUM-0016 TCB-007DBD18 RET-832013EA TIME-09 11 32 8571084384 INTERVAL-00.0004786875    =000002=
                1-0000  C4C2F240 60405CC5 E7C35C                                          *DB2 - *EXC*                    *
                2-0000  E7C3F0F5 00000C10                                                 *XC05....                       *
                3-0000  E3C5E2E3 C3F0F540                                                 *TESTC05                        *
                4-0000  0311F46C                                                          *..4%                           *
                5-0000  00000000                                                          *....                           *
                6-0000  00000000                                                          *....                           *
                7-0000  C4E2D5F2 D3D6E340 000B0000 0001CD98  00000000 00000000 E2E8E2C1 C4D44040  *DSN2LOT .......q........SYSADM  *
                  0020  40404040 40404040 E7C3F0F5 00000000  00000000 00000000 00000000 C4C2F2D5  *        XC05................DB2N*
                  0040  C5E34040 E3F1F3F0 F3404040 18B370D1  3B050000 00010004 00000C10 00000000  *ET  T1303   ...J.................*
                  0060  00000000 00000000 E7C3F0F5 00000C10  02F10368 00000000 0311F494 00040000  *........XC05.....1........4m....*
                  0080  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                  00A0  E3C5E2E3 C3F0F540 01C00000 00000000  00000000 00000000 00000000 00000000  *TESTC05 ........................*
                  00C0  00000000 0000                                                     *......                         *
```

Figure 30. Exception trace entries

Changed DB2 message number prefix

DSN2nnn messages (for example, DSN2007) are issued for the CICS/ESA 4.1 CICS DB2 attachment facility, equivalent to the DSNCnnn messages (for example, DSNC007) issued for the earlier DB2 attachment facility.

CICS DB2 attachment facility fails to initialize

If the CICS DB2 attachment facility does not initialize, this may be because:

- The CICS DB2 attachment facility level is incorrect
- The resource control table level is incorrect
- DB2 subsystem ID is incorrectly specified

CICS DB2 attachment facility level is incorrect: If there is a mismatch between the CICS release and the CICS DB2 attachment facility level either of the following DB2 messages can be issued:

- DSN2056, if a CICS/ESA 4.1 CICS DB2 attachment facility attempts to connect to CICS/ESA 3.3 or an earlier release.

  In this case, make sure that the CSD definition of the DSNC transaction specifies program DSNCCOM1, and the only entry in the CICS PLTPI for the CICS-DB2 interface is for program DSNCCOM0.

- Message DSNC057, if CICS/ESA 4.1 attempts to connect to a a DB2-supplied attachment facility.

  In this case, make sure that the CSD definition of the DSNC transaction specifies program DSN2COM1, and the only entry in the CICS PLTPI for the CICS-DB2 interface is for program DSN2COM0.

Resource control table level is incorrect: If there is a mismatch between resource control table and the CICS DB2 attachment facility being used:

- Message DSNC059I indicates that the wrong macro library was used to assemble DSNCRCT, the resource control table for CICS/ESA 3.3 and earlier.

  Reassemble DSNCRCT using the DSNCRCT macro shipped in the DB2 macro libraries.

- Message DSN2059I indicates that the wrong macro library was used to assemble DSN2CT, the resource control table used for CICS/ESA 4.1.

  Reassemble DSN2CT using the DSNCRCT macro shipped in the CICS macro libraries.

- Message DSN2002I indicates that the resource control table DSN2CT with suffix xx could not be loaded.

  – Reenter the start command with the correct suffix.

  – Ensure that the resource control table DSN2CT is in the correct application program library, which is concatenated in the JOBLIB or STEPLIB statement of your CICS startup JCL.

    If DSN2CT is not in your application program library, ensure that you are using the correct version of the DSNCRCT macro to assemble DSN2CT. For CICS/ESA 4.1, the correct DSNCRCT macro is in the CICS macro library rather than in the DB2 macro library.

DB2 subsystem ID incorrectly specified:

- Message DSN2060I indicates that the DSN2STRT INITPARM specified on the CICS initialization job is incorrect.

  The CICS DB2 attachment facility attempts to initialize with the subsystem ID specified in the DSNC STRT command. If no subsystem ID is provided there, it attempts to initialize with the subsystem ID specified in the resource control table.

  If you want to override the subsystem ID, restart the attachment facility using the DSNC STRT command with the correct subsystem ID. Alternatively, you can correct the DSN2STRT INITPARM and reinitialize CICS. See “Connecting DB2 to CICS automatically” on page 444 for information on specifying the DB2 subsystem identifier.

- Message DSN2061I indicates that no resource control table suffix was specified on the DSNC STRT command, and the format of the DSN2STRT INITPARM on the CICS initialization job is incorrect. (See “Connecting DB2 to CICS automatically” on page 444 for the correct format.)

  To start the CICS DB2 attachment facility, use the DSNC STRT command and specify a resource control table suffix on the command. Alternatively, you can correct the DSN2STRT INITPARM and reinitialize CICS.

DBCTL operator transaction, CDBM

In CICS/ESA 4.1, a CICS-supplied transaction, CDBM, enables you to issue IMS commands from CICS to DBCTL from a CICS terminal. Using CDBM offers a similar operator interface to the one provided by CEMT DLIDATABASE, which is used with CICS and local DL/I.

Benefits

CDBM offers an improved operator interface to DBCTL, and enables system administrators that don't have access to an MVS console to issue DBCTL commands from a CICS terminal.

Changes to CICS externals

CDBM causes the following changes to CICS externals:

- Changes to DBCTL system definition
- Changes to DBCTL operations
- Changes to CICS-supplied transactions
- Changes to resource definition online (RDO)
- Changes to security
- Changes to messages

Changes to DBCTL system definition

CICS/ESA 4.1 provides a transaction, CDBM, which enables DBCTL operator commands to be issued at a CICS terminal, as described in "DBCTL operator transaction, CDBM." CDBM uses the AIB commands (available from IMS/ESA 5.1 onwards) that can be issued across the DRA interface between CICS and DBCTL. For more information, see "Issue IMS AIB call format" on page 458.

To use CDBM you must:

1. Have a DBCTL system running IMS/ESA 5.1, or later.
2. Generate, and add to the DBCTL system, a PSB named DFHDBMP. We recommend that you specify parallel scheduling in this PSB, to enable multiple CDBM transactions to be active at the same time. DFHDBMP need not have any associated PCBs. Example input for the PSBGEN is:

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

   The IOASIZE parameter must be large enough to cope with the largest AOI command issued. Large AOI commands can result from using wild cards. For example, issuing CDBM /START DATABASE D* results in a start command being issued for all database names beginning with D. See the *IMS/ESA Utilities Reference: System* manual, SC26-4629, for information on defining IOASIZE.

Changes to DBCTL operations

If you are using CICS/ESA 4.1 with IMS/ESA 5.1 or later, you can choose to issue operator commands via a CICS-supplied transaction, CDBM, or via an MVS console as in previous releases.

You can use CDBM to issue most of the IMS operator commands that are valid for DBCTL across the DRA interface to DBCTL to display and change the state of selected resources. When dealing with databases, you can use an asterisk (*) to

refer to generic groups. For example DI21* refers to all databases starting with the characters DI21.

The syntax for issuing these commands is as follows:

```
CDBM /xxxxxxxx
```

where / is the default command recognition character (CRC) and xxxxxxxx is a valid IMS operator command. Note that the default CRC must be used because the command is sent to whichever DBCTL system is currently connected to CICS, which may or may not be the one you are using.

The following IMS operator commands are valid with CDBM:

- /CHANGE
- /CHECKPOINT (simple form) and /CHECKPOINT STATISTICS
- /DBDUMP
- /DBRECOVERY
- /DELETE
- /DEQUEUE
- /DISPLAY
- /LOCK
- /LOG
- /PSTOP
- /RMCHANGE
- /RMDELETE
- /RMGENJCL
- /RMINIT
- /RMLIST
- /RMNOTIFY
- /START
- /STOP
- /SWITCH OLDS
- /TRACE SET PI
- /UNLOCK
- /VUNLOAD

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

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

For more information, see the *CICS/ESA CICS-IMS Database Control Guide*.

Changes to CICS-supplied transactions

You can use the new CDBM transaction to issue DBCTL operator commands from a CICS terminal. CDBM is applicable only to CICS systems using IMS/ESA Database Control (DBCTL) interface with IMS/ESA 5.1 or later.

You issue CDBM from a CICS terminal. The syntax of the CDBM transaction is:

```
CDBM /DBCTL operator command
```

where / is the default command recognition character (CRC) for DBCTL. Note that for CDBM the default CRC is the one that you must use.

You can also start the transaction by typing the identifier, CDBM, on the command line of your display, as follows:

```
CDBM
```

Press the ENTER key, and CICS returns the display shown in Figure 31.

```
CDBM                 CICS-DBCTL Operator Transaction                 93.335
                                                                    13:24:20

Type IMS command.
             ____________________________________________________________
             ____________________________________________________________
             ____________________________________________________________
             ____________________________________________________________


For /DBDUMP or /DBRECOVER commands

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


Press enter to display responses.


CICS APPLID DBDCCICS
DBCTL ID    SYS3


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

Figure 31. CICS-DBCTL Operator Transaction

Pressing PF1 displays the following help screen for CDBM:

```
 CDBM               Help: CICS-DBCTL Operator Transaction


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

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

Responses      Use the PF keys to page IMS responses.

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

End of volume  For /DBDUMP or /DBRECOVER commands only
               Choose one.
                 1. Do not force end of volume
                 2. Force end of volume

CICS APPLID    These are shown for information.
DBCTL ID

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


F3=Exit  F12=Cancel
```

Figure 32. CICS-DBCTL Operation Transaction Help panel

Responses to DBCTL commands issued using CDBM are returned in a screen like the one shown in Figure 33, which shows the first of a number of screens issued in response to a /DISPLAY DB ALL command.

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

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

Figure 33. CICS-DBCTL IMS Responses

For more information, see the *CICS/ESA CICS-IMS Database Control Guide* and the *CICS/ESA CICS-Supplied Transactions*.

Changes to resource definition

CDBM causes the following changes to group DFHDBCTL in DFHLIST.

- DFHDBMP is added to the list of programs
- DFHDBDE is added to the list of map sets
- CDBM is added to the list of transactions.

For more information, see the *CICS/ESA Resource Definition Guide*.

Changes to security

We recommend that you add the CDBM transaction to the list of transactions defined in category 2. This category includes transactions that are either initiated by the terminal user or are associated with a terminal, and are subject to resource and command security checks. If you do not want to give all users access to this transaction, we recommend that you define it in a subcategory; for example, one for system administrators.

For more information, see the *CICS/ESA CICS-RACF Security Guide*.

Changes to CICS messages

There are new messages added, DFHDB8228 through DFHDB8239, relating to the CDBM transaction. For details of these and other new messages, see the *CICS/ESA Migration Guide*.

Specifying a DBCTL system identifier (DBCTLID)

If you are running CICS and DBCTL in a sysplex, you need a separate, uniquely named, DBCTL subsystem on each MVS image that supports DL/I data sharing. The application-owning regions running the DL/I transactions that access the shared data must connect to the correct DBCTL by the name specified on the DBCTLID parameter. In earlier releases, the DBCTLID can be specified only in the DRA startup table that is named in the CDBC connection command or CICS initialization parameter. This means that a separate DRA table is required for each DBCTL. In many cases the DBCTLID is the only parameter that is different between one DRA startup table and another.

In CICS/ESA 4.1, you can override the DBCTLID specified in the DRA startup table using one of the following:

- The INITPARM system initialization parameter
- The CDBC CONNECT panel
- The CDBC CONNECT line command.

Benefits

The ability to specify a unique DBCTL identifier in CICS/ESA 4.1 makes it possible for a single generic DRA startup table (DFSPZPxx) to be used for all equivalent DBCTL instances in a SYSPLEX. This removes the administrative effort needed to create and maintain multiple tables.

Changes to CICS externals

Specifying a DBCTL identifier in CICS/ESA 4.1 causes the following changes to CICS externals:

- Changes to system definition
- Changes to operations
- Changes to CICS-supplied transactions
- Changes to security

Changes to system definition

The CICS system initialization parameter INITPARM is extended, as follows:

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

where *xx* is a 1- to 2-character DRA startup table suffix, which must be entered if you specify INITPARM, and *yyyy* is an optional 1- to 4-character DBCTL identifier. Specifying a DBCTL identifier on INITPARM overrides the value in the DRA startup table parameter, DBCTLID.

For more information, see the *CICS/ESA System Definition Guide*.

Changes to operations

To connect CICS automatically to a specific DBCTL, add an entry for the DBCTL connection program, DFHDBCON to the PLTPI, as in previous releases. You then use the CICS system initialization parameter, INITPARM, as described in "Changes to system definition."

To connect CICS to a specific DBCTL using CICS sequential device support for simulated terminals (CRLP) enter the following:

```
CDBC CONnect SUFfix(xx) DBCtlid(yyyy)\
```

where *xx* is the 1- to 2-character DRA startup table suffix and *yyyy* is the 1- to 4-character DBCTL identifier, both of which are optional. Specifying a DBCTL identifier overrides the value in the DRA startup table parameter, DBCTLID.

For more information, see the *CICS/ESA System Definition Guide*.

Changes to CICS-supplied transactions

A field on the CDBC panel enables you to use the CDBC transaction to specify a DBCTLID to override the one in the DRA startup table. The syntax for connection to DBCTL via CDBC is:

```
CDBC CONnect [SUFfix(xx)] [DBCtlid(yyyy)]
```

Typing CDBC on a 3270-type terminal displays the screen shown in Figure 34 on page 457 for connecting CICS to, and disconnecting it from, DBCTL.

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

          Select one of the following:

               1 Connection
               2 ORDERLY disconnection
               3 IMMEDIATE disconnection


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

  DFHDB8209D DBCTL orderly disconnection requested. Press PF5 to confirm.

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


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

Figure 34. CDBC transaction menu screen

The following help screen can be displayed for the CDBC transaction:

```
                HELP : CICS-DBCTL CONNECTION/DISCONNECTION

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

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

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

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

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

                 PRESS ENTER TO RETURN TO SELECTION SCREEN
```

Figure 35. CDBC transaction menu help screen

For more information, see the *CICS/ESA CICS-Supplied Transactions*.

Release DBCTL threads at syncpoint

Each CICS transaction that accesses DBCTL uses a thread that is not released until the transaction terminates. The maximum number of threads that can be used concurrently in a single DBCTL is 255. Pseudoconversational transactions terminate and restart, thus releasing threads relatively quickly. However, conversational transactions hold onto resources, whether they are using them or not, until they terminate when the task is completed. This means that CICS-DBCTL cannot support more than 255 concurrently conversational transactions whether they are active or not. In CICS/ESA 4.1, CICS-DBCTL threads are released at syncpoint and are not reacquired until the next request is issued to IMS.

Benefits

Releasing DBCTL threads at syncpoint means that the limit on the number of conversational transactions is removed, and all resources are made available for reuse promptly. This provides greater flexibility if conversational programs are needed.

Note that you do not need to make any changes to your existing applications in order to benefit from this, and there are no changes to CICS externals.

For more information, see the *CICS/ESA CICS-IMS Database Control Guide*.

Issue IMS AIB call format

If you are using CICS/ESA 4.1 with IMS/ESA 5.1 or later, a CICS transaction can issue the DL/I calls GMSG, ICMD, and RCMD in AIB format. Note that AIB applies to Call DL/I only, not EXEC DLI, and the calls cannot be function shipped. (Programming interface information on these calls is in the *IMS/ESA Application Programming: DL/I Calls* manual, SC26-3062.)

Benefits

This enables DBCTL operator commands to be sent in a CICS transaction, CDBM. (See “DBCTL operator transaction, CDBM” on page 451.) Operations such as bringing IMS databases online can thus be performed using CICS, instead of via an MVS console.

Changes to CICS externals

Issuing DL/I calls in AIB format causes changes to CICS abends.

A CICS abend, ADPM, is issued if a program attempts to issue any DL/I call other than GMSG, ICMD, or RCMD in AIB format.

Application programming considerations

Information on defining AIB format instead of PCB format and on the AIBTDLI entry point for link-edit is in the *IMS/ESA Application Programming: DL/I Calls* manual, SC26-3062.

Removal of support for IMS/VS 2.2 with local DL/I

Support for IMS/VS 2.2 via the local DL/I interface is withdrawn in CICS/ESA 4.1. (Note that this applies to the local DL/I interface only, as support for DBCTL was not available prior to IMS/ESA Version 3.)

For more information, see the *CICS/ESA Migration Guide* and the *CICS/ESA CICS-IMS Database Control Guide*.

Changes to CICS externals

The removal of support for IMS/VS 2.2 via local DL/I causes the following changes to CICS externals:

- Changes to installation
- Changes to system definition

Changes to installation

Note that you can no longer specify IMS/VS 2.2 on the DFHISTAR installation job.

For more information, see the *CICS/ESA Installation Guide*.

Changes to system definition

If you specify IMS/VS 2.2 during local DL/I system generation, the PDIR and DDIR tables will not assemble, and the following local DL/I modules fail with an MNOTE message:

DFHDLDBD, DFHDLILP, DFHDLPSB, DFHDLQ, DFHDLR, and DFHDLX.

For more information, see the *CICS/ESA System Definition Guide*.

Resource manager interface global user exits, XRMIIN and XRMIOUT

Product-sensitive programming interface

Two global user exits are added to the AP domain to enable you to monitor activity across the resource manager interface. XRMIIN is invoked just before control is passed from the resource manager interface (RMI) to a task-related user exit. XRMIOUT is invoked just after control is passed back to the RMI.

For information about using these exits, see the *CICS/ESA Customization Guide*.

Benefits

Using XRMIIN and XRMIOUT, you can monitor RMI activity. For example, you can monitor control being passed to and from DFHEDP for EXEC DLI requests, DFHDBAT for DBCTL requests, or DSNCSQL for DB2 commands.

Changes to CICS externals

There are two new global user exits, XRMIIN and XRMIOUT, for the resource manager interface (RMI).

Table 37. Global user exits

Program or domain	Exit name	Where or when invoked
Resource manager interface (DFHERM)	XRMIIN	Before execution of an EXEC DLI, EXEC SQL or RMI command.
Resource manager interface (DFHERM)	XRMIOUT	After execution of an EXEC DLI, EXEC SQL or RMI command.

Exit XRMIIN

Invoked before control is passed to a task-related user exit program.

Exit-specific parameters

UEPTRUEN Address of the name of the task-related user exit program.

UEPTRUEP Address of the parameter list to be passed to the task-related user exit program. See note.

UEPRECUR Address of usage recursion count

Note: The task-related user exit program's parameter list is mapped by a DFHUEPAR DSECT that shares common field names with the global user exit program's DFHUEPAR parameter list. To include both DSECT definitions in your exit program, you must code:

```
DFHUEXIT TYPE=EP,ID=XRMIIN
DFHUEXIT TYPE,TYPE=RM
```

The statements must be coded in this order.

The two DFHUEPAR parameter lists, the global user exit's and the task-related user exit's, occupy separate areas of storage. The task-related user exit's parameter list is provided for information only; you should not amend it in any way.

Return codes

UERCNORM Continue processing.

UERCPURG Task purged during XPI call.

XPI calls

All can be used.

API commands

All can be used. However, CALLDLI, EXEC DLI, or EXEC SQL commands must **not** be used.

Exit XRMIOUT

	Invoked after control has returned from a task-related user exit program.	
Exit-specific parameters	**UEPTRUEN**	Address of the name of the task-related user exit program.
	UEPTRUEP	Address of the parameter list passed to the task-related user exit program. See note.
	UEPRECUR	Address of usage recursion count

	Note:	The task-related user exit program's parameter list is mapped by a DFHUEPAR DSECT that shares common field names with the global user exit program's DFHUEPAR parameter list. To include both DSECT definitions in your exit program, you must code:
		`DFHUEXIT TYPE=EP,ID=XRMIOUT`
		`DFHUEXIT TYPE,TYPE=RM`
		The statements must be coded in this order.
		The two DFHUEPAR parameter lists, the global user exit's and the task-related user exit's, occupy separate areas of storage. The task-related user exit's parameter list is provided for information only; you should not amend it in any way.

Return codes	**UERCNORM**	Continue processing.
	UERCPURG	Task purged during XPI call.
XPI calls	All can be used.	
API commands	All can be used. However, CALLDLI, EXEC DLI, or EXEC SQL commands must **not** be used.	

Note: These global user exit points are for information only; do not change the parameters in the list in any way.

Recursion warning: You must take care if your exit program issues calls to other external resource managers that use the RMI, as this causes recursion, and a loop can ensue. It is your responsibility to avoid entering a loop, and you should use UEPRECUR for this purpose. However, use of the RMI from within the exit program is not recommended.

For information about using these exits, see the *CICS/ESA Customization Guide*.

End of Product-sensitive programming interface

DBCTL installation verification procedure

CICS/ESA 4.1 provides an installation verification procedure (DFHIVPDB) for CICS-DBCTL.

Changes to CICS externals

CICS/ESA 4.1 provides an installation verification procedure (IVP) for CICS-DBCTL, called DFHIVPDB. DFHIVPDB connects CICS to DBCTL using the sample DRA startup table with a suffix of IV. The DBCTL system it connects to is called IVP3, which is generated by the IMS IVP. The IMS IVP also builds the sample database DI21PART, and issues the necessary PSBGEN and ACBGEN jobs for use by the DBCTL system. The IMS IVP also builds a DRA startup table called DFSPZPIV which is placed in IMS.RESLIB. The CICS job includes IMS.RESLIB in the STEPLIB concatenation to load the table.

For more information, see the *CICS/ESA Installation Guide*.

Chapter 28. Changes for application programming

This chapter describes a number of changes in CICS/ESA 4.1 that affect application programming. These items are:

- Four-digit year numbers for dates in the 21st century
- Removal of file control command-level restrictions
- New options on the ERASE parameter
- The PARTNER option for APPC commands
- | Add NAME parameter to EXEC CICS WAIT EVENT, WAIT EXTERNAL, WAITCICS
- + Add MAPNAME and MAPSETNAME options on INQUIRE and SET TERMINAL
- # BMS global user exits
- Prevent translator producing CBL cards
- CEDF enhancements
- + C++ support.

This chapter contains General-use Programming Interface information.

Four-digit year numbers for dates in the 21st century

This section outlines the changes in CICS/ESA 4.1 to handle dates in the 21st century. These changes are to enable CICS users and programs to recognize that after 31 December 1999 the dates start from 1 January 2000.

With dates represented by a two-digit year value, the change of century causes the date to change from 99 to 00. Where this is thought not significant (such as in CICS screens that show the date as YY.DDD) the date format is not changed. Otherwise, the date is made available with a century indicator, as for the MVS date facility. CICS maintains the date in the form 0CYYDDD in the CSA and EIB (where C=century number minus 19, YY=year of century, and DDD=day of year), and converts it to the standard you specify for display.

These changes also add new options to the EXEC CICS FORMATTIME command, and the EXEC CICS INQUIRE VOLUME command, to specify dates with four-digit year values. These changes are outlined below:

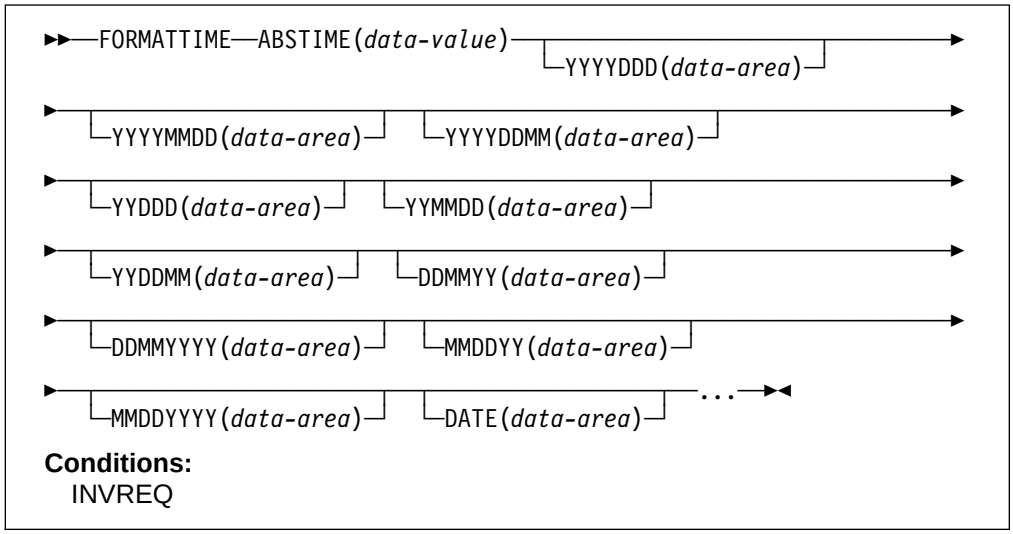
Changes to application programming

The following application programming commands are changed to handle four-digit year values, as outlined in this section:

- EXEC CICS FORMATTIME command, to add the new options YYYYDDD, YYYYMMDD, YYYYDDMM, DDMMYYYY, and MMDDYYYY
- EXEC CICS INQUIRE VOLUME command, to add the option LONGDATE(data-area) option

EXEC CICS FORMATTIME

The syntax for the new options of the EXEC CICS FORMATTIME command is:



INQUIRE VOLUME LONGDATE(data-area)

The INQUIRE VOLUME command, which applies only to standard-labeled TAPE journals, retrieves information about a named volume. The option LONGDATE(data-area) is added, to return a date of the form YYYYDDD. This option is an alternative to the DATE(data-area) option.

Removal of CICS file control update restrictions

The processing of file control requests in earlier releases includes checks to prevent a transaction from issuing requests such as READ for UPDATE, REWRITE, UNLOCK, and DELETE to a given file before a previous UPDATE request to the same file has been completed. These checks prevent the transaction becoming deadlocked.

There is no longer any need for these restrictions because of deadlock resolution introduced in CICS/ESA Version 3, and changes added by VSAM, and the restrictions are removed for *local* files. This means that CICS/ESA 4.1 permits multiple concurrent update requests from the same transaction.

BDAM files

For BDAM files, it is possible to perform concurrent updates of records within the same block, but this is not advisable as some of the updates would be lost when doing this. For this reason, you should not specify SERVREQ=NOEXCTL on file resource definition for any BDAM file which may be used in this way.

Deadlock checking

CICS does not attempt to detect deadlock situations for READ for UPDATE requests that use the keyword TOKEN.

Benefits

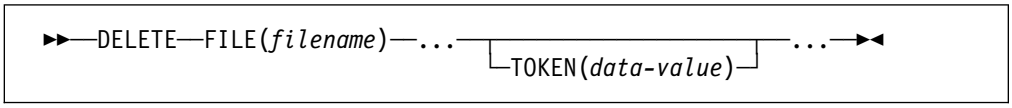
The deadlock checks in the earlier releases required a redesign of the application program logic when migrating from macro- to command-level. Removing the deadlock checks allows you to migrate your macro-level applications to command-level with minimal design change.

Changes to the application programming interface

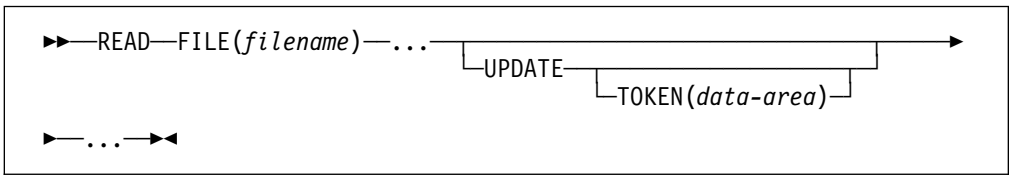
A unique identifier identifies each request. These identifiers take the form of a TOKEN keyword for each of READ for UPDATE commands. This token is a 4-character value that is unique within a particular task and can be returned with the associated REWRITE, DELETE, or UNLOCK command, and therefore binds the two requests together. It is also possible to have a series of multiple UPDATES at the same time so long as each UPDATE is defined with its own unique token. You can also have a single UPDATE request within such a data set that does not have a token.

Note: There is no function-shipping support for file control requests that contain the TOKEN keyword, therefore you cannot use the TOKEN option for remote files.

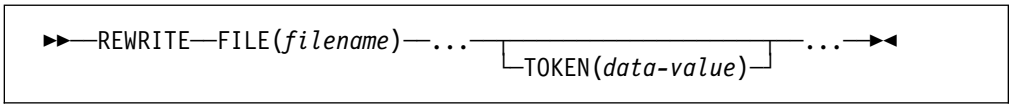
The TOKEN keyword is added to the following commands:



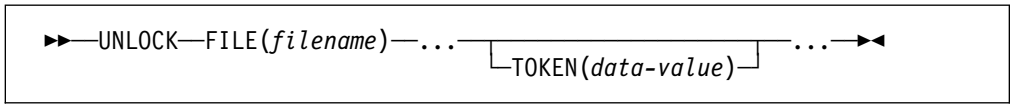
TOKEN(data-value)
specifies as a fullword binary value, a unique request identifier for a DELETE, and is used to associate it with a previous READ for UPDATE request. This is an input value returned by the task to file control.



TOKEN(data-value)
specifies as a binary value a unique request identifier for a READ, used to control multiple UPDATE operations on a data set. This is an output value provided by file control to the requesting task.



TOKEN(data-value)
specifies, as a binary value, a unique request identifier for a REWRITE, used to associate it with a previous READ UPDATE request.



TOKEN(data-value)
specifies, as a binary value, a unique request identifier for an UNLOCK, used to associate it with a previous READ for UPDATE request.

Exceptional conditions for the token option

DELETE

INVREQ
INVREQ occurs under the following conditions:

- A DELETE instruction includes a token whose value cannot be matched against any token in use for an existing READ for UPDATE request (RESP2=47).
- An attempt is made to function-ship a request which includes a TOKEN keyword (RESP2=48).

READ

INVREQ
An attempt is made to function-ship an UPDATE request that includes a TOKEN keyword (RESP2=48).

REWRITE

INVREQ
INVREQ occurs under the following conditions:

- If a REWRITE instruction includes a token whose value cannot be matched against any token in use for an existing READ for UPDATE request (RESP2=47).
- If a REWRITE command is issued without a token, then file control attempts to bind either of these with a previous READ for UPDATE command also without a token and it cannot be found (RESP2=30).
- An attempt is made to function-ship a request which includes a TOKEN keyword (RESP2=48).

UNLOCK

INVREQ
occurs in any of the following situations:

- An unlock includes a token whose value cannot be matched against any token in use for an existing READ for UPDATE request (RESP2=47).
- An attempt is made to function-ship a request which includes a TOKEN keyword (RESP2=48).

If an UNLOCK command does not have a token, an attempt is made to match it to a READ for UPDATE which also does not have a token. If this is not found, no action is taken and a NORMAL response is returned.

New options on the ERASE parameter

The options DEFAULT|ALTERNATE have been added to the ERASE parameter on the SEND, CONVERSE, SEND TEXT, and SEND CONTROL commands.

The ERASE parameter specifies that the buffer or screen is to be erased and the cursor returned to the upper left corner of the screen before writing occurs.

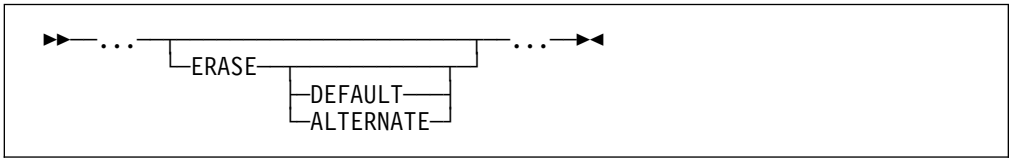
In earlier releases, the screen size remains unchanged from its previous setting on the transaction PROFILE, or the default screen size if the CLEAR key is pressed.

You can now change the screen size within the application using the DEFAULT or ALTERNATE options with ERASE. This setting applies until changed again with another ERASE parameter. If the terminal cannot use an ALTERNATE screen size, then ALTERNATE is ignored, which is consistent with the initial setting from the profile.

Changes to the application programming interface

The ERASE parameter has the options DEFAULT|ALTERNATE added to the following commands:

- CONVERSE (LUTYPE2/LUTYPE3)
- CONVERSE (3270 display)
- CONVERSE (3270 logical)
- CONVERSE (3650-3270)
- CONVERSE (3790 3270-display)
- SEND (LUTYPE2/LUTYPE3)
- SEND (3270 display)
- SEND (3270 logical)
- SEND (3650-3270)
- SEND (3790 3270-display)
- SEND (3790 3270-printer)
- SEND CONTROL
- SEND MAP
- SEND TEXT
- SEND TEXT NOEDIT



ALTERNATE
set the terminal to use the ALTERNATE screen size.

DEFAULT
set the terminal to use the DEFAULT screen size.

ERASE(data-value)
specifies that the screen printer buffer or partition is to be erased and the cursor returned to the upper left corner of the screen. (This option applies only to the 3270, or 8775, and to the 3604 Keyboard Display.) The first output operation in any transaction, or in a series of pseudoconversational transactions, should always specify ERASE. For transactions attached to 3270 screens or printers, unless explicitly overridden by the DEFAULT or ALTERNATE option, this also ensures that the correct screen size is selected, as defined for the transaction by the SCRNSZE option in the profile definition.

PARTNER option on EXEC CICS APPC conversations

The PARTNER option is added to the following EXEC CICS commands:

- ALLOCATE
- CONNECT PROCESS
- GDS ALLOCATE
- GDS CONNECT PROCESS

It is for use in a distributed transaction processing (DTP) environment, where programs in different systems interact with each other across APPC (LUTYPE6.2) links. The PARTNER option specifies a PARTNER resource definition, which identifies a remote logical unit, remote transaction program, and a communication profile to be used on the session.

Benefits

Use of the PARTNER option makes it unnecessary to name the partner LU and transaction explicitly on EXEC CICS ALLOCATE and CONNECT PROCESS commands. Instead, the details of each partner program can be contained in a single definition. This makes your DTP programs more maintainable.

The introduction of the PARTNER option for APPC conversations makes available to the EXEC CICS application programming interface (API) a resource previously available only to the CPI Communications API.

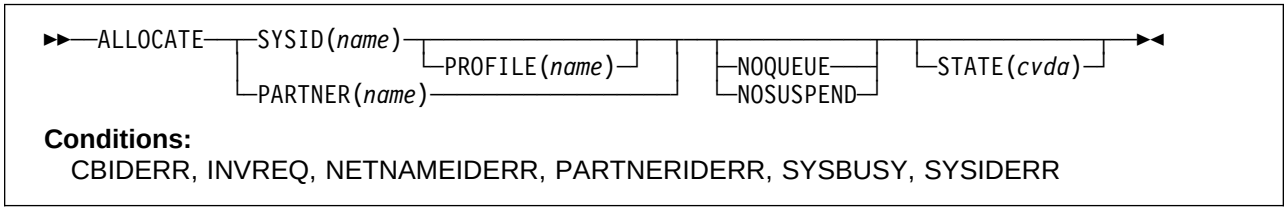
EXEC CICS ALLOCATE (APPC mapped)

The PARTNER option is added to this command.

Function

Acquire a session to a remote APPC logical unit for use by an APPC mapped conversation.

Syntax



Conditions:
CBIDERR, INVREQ, NETNAMEIDERR, PARTNERIDERR, SYSBUSY, SYSIDERR

ALLOCATE (APPC mapped) options

PARTNER(name)

specifies the name (8 characters) of a set of definitions that include the names of a remote LU (NETNAME) and a communication profile to be used on the allocated session. You can use this option as an alternative to specifying SYSID and PROFILE explicitly.

ALLOCATE (APPC mapped) conditions

PARTNERIDERR

occurs if the name specified in the PARTNER option is not recognized by CICS.

Default action: terminate the task abnormally.

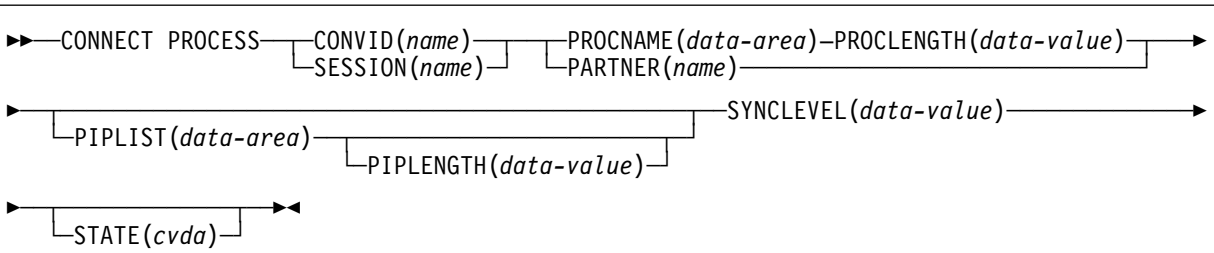
EXEC CICS CONNECT PROCESS

The PARTNER option is added to this command.

Function

Initiate an APPC mapped conversation.

Syntax



Conditions:
INVREQ, LENGERR, NOTALLOC, PARTNERIDERR, TERMERR

CONNECT PROCESS options

PARTNER(name)

specifies the name (8 characters) of a set of definitions that includes the name (or extended name) of a remote partner transaction (TPNAME or XTPNAME). You can use this option as an alternative to PROCNAME and PROCLENGTH.

CONNECT PROCESS conditions

PARTNERIDERR

occurs if the name specified in the PARTNER option is not recognized by CICS.

Default action: terminate the task abnormally.

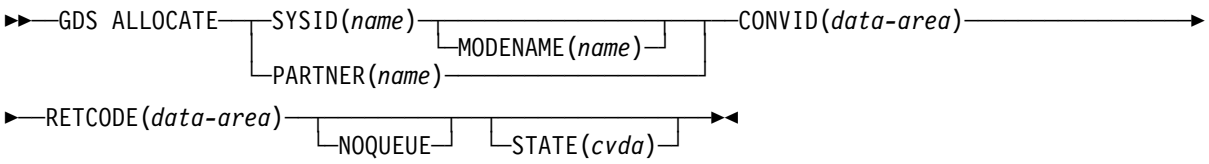
EXEC CICS GDS ALLOCATE

The PARTNER option is added to this command.

Function

Acquire a session to a remote APPC logical unit for use by an APPC basic conversation (assembler-language and C/370 programs only).

Syntax



Conditions:

EXEC CICS conditions are never raised on GDS commands. The return code is given in RETCODE (see table following the option descriptions). For a full list of return code values, see the *CICS/ESA Distributed Transaction Programming Guide*.

GDS ALLOCATE options

MODENAME(name)

specifies the name of the mode group from which the session is to be acquired. If you specify SYSID and omit MODENAME, CICS selects a modename from those defined for the system.

PARTNER(name)

specifies the name (8 characters) of a set of definitions that include the names of a remote LU (NETNAME) and a communication profile to be used on the allocated session. For APPC basic conversations, the only relevant attribute set by the profile is MODENAME.

If you use this option as an alternative to SYSID and MODENAME, CICS uses the NETNAME and MODENAME from the PARTNER definition.

GDS ALLOCATE return codes

RETCODE (hexadecimal)	Description
01 0C 14	The NETNAME specified in the PARTNER definition is not known.
02 0C 00	PARTNER is not known.
06 00 00	The PROFILE specified in the PARTNER definition is not known.

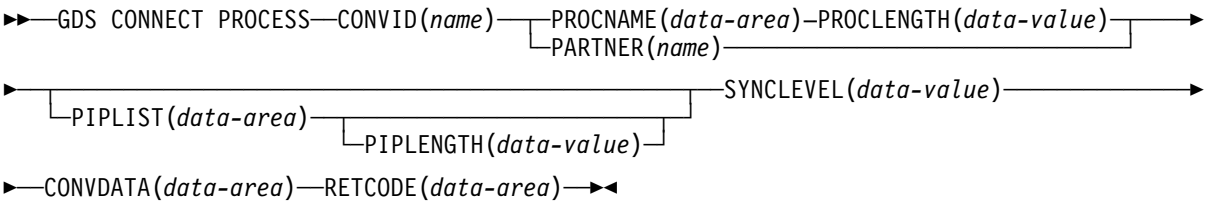
EXEC CICS GDS CONNECT PROCESS

The PARTNER option is added to this command.

Function

Initiate an APPC basic conversation (assembler-language and C/370 programs only).

Syntax



Conditions:
EXEC CICS conditions are never raised on GDS commands. The return code is given in RETCODE (see table following the option descriptions). For a full list of return code values, see the *CICS/ESA Distributed Transaction Programming Guide*.

GDS CONNECT PROCESS options

PARTNER(name)
specifies the name (8 characters) of a set of definitions that includes the name (or extended name) of a remote partner transaction (TPNAME or XTPNAME). You can use this option as an alternative to PROCNAME and PROCLENGTH.

GDS CONNECT PROCESS return codes

RETCODE (hexadecimal)	Description
02 0C 00	PARTNER is not known.

New response and abend codes

Table 38 lists the new response codes that may be returned in the EIBRCODE field of the EXEC interface block (EIB), following an EXEC CICS ALLOCATE or CONNECT PROCESS command.

Table 38. CICS response codes for ALLOCATE or CONNECT PROCESS (mapped)

EIBFN	EIBRCODE	Condition
04..	EC	PARTNERIDERR
04..	ED	NETNAMEIDERR

There is one new abend code, AEX7, meaning that the NETNAMEIDERR condition was not handled by your program. For further details, see the description of abend AEIA in the *CICS/ESA Messages and Codes* manual.

Defining remote resources for DTP

In CICS/ESA 3.3, you need to create PARTNER definitions only if you used the CPI Communications API on APPC conversations. In CICS/ESA 4.1, you can use the PARTNER resource with the CICS API also on APPC conversations. If you do so, you may need to create new PARTNER definitions, unless you can use existing ones defined for CPI Communications programs.

Add NAME parameter to EXEC CICS WAIT EVENT, WAIT EXTERNAL, WAITCICS

When a transaction is routed from a terminal-owning region to an application-owning region, the task in the terminal-owning region waits while the transaction is running in the AOR. You can determine the reason for the WAIT using the CEMT INQUIRE TASK or EXEC CICS INQUIRE TASK command.

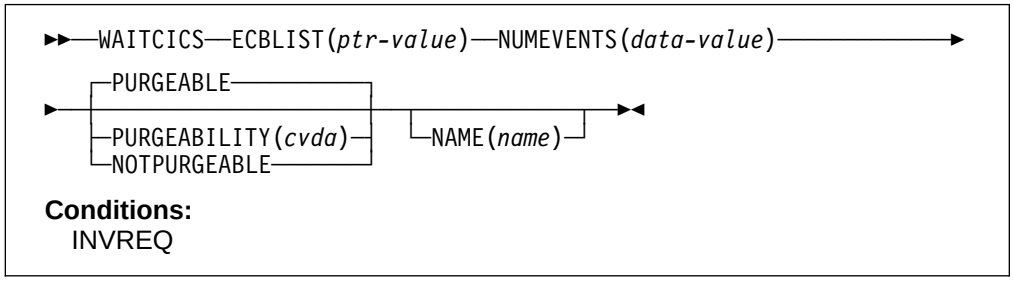
CICS/6000 operates in a TOR and routes transactions elsewhere. As well as waiting on the MRO or ISC link, CICS/6000 makes extensive use of EXEC CICS WAIT EVENT and EXEC CICS WAIT EXTERNAL when routing a transaction across the network.

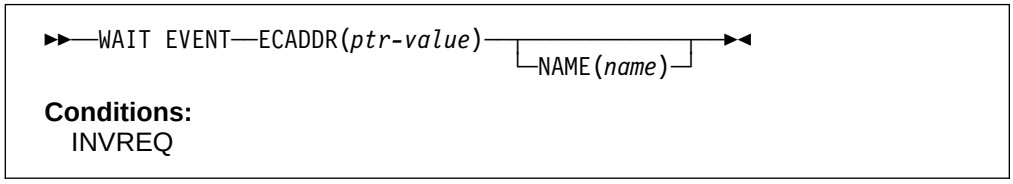
To facilitate problem determination, CICS/6000 needs to be able to specify on the command the reason for the WAIT, and have this reason displayed by CEMT.

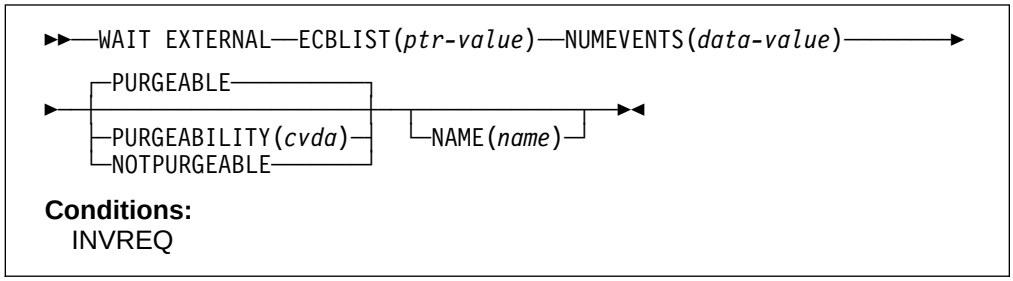
An 8-character reason for the wait, in the NAME parameter, is added to EXEC CICS WAITCICS, EXEC CICS WAIT EVENT, and EXEC CICS WAIT EXTERNAL commands.

Changes to the application programming interface

An 8-character parameter, NAME, is added to EXEC CICS WAITCICS, EXEC CICS WAIT EVENT, and EXEC CICS WAIT EXTERNAL commands.







Name option on the wait commands

NAME(name)

specifies the symbolic name, 1–8 alphanumeric characters, as the reason for the wait. The value you specify is returned in the SUSPENDVALUE or HVALUE option respectively of the EXEC CICS INQUIRE TASK or a CEMT INQUIRE TASK commands.

Problem determination

Table 39. Resources that a suspended task might be waiting on

Resource type	Resource name	Suspending module	DSSR call	Task	Where to look next
EKCWAIT	SINGLE/NONE	DFHEKC	WAIT_OLDW	User	See the *CICS/ESA Problem Determination Guide*
USERWAIT	NAME	DFHEIQSK	WAIT_MVS WAIT_OLDW	User	See the *CICS/ESA Problem Determination Guide*

+ MAPNAME and MAPSETNAME options on INQUIRE and SET TERMINAL

APAR PN69050

Documentation for PN69050 added on 10 July 1996

MAPNAME and MAPSETNAME options allow CICS programs to INQUIRE and SET the names of the most recently used map and mapset used in BMS SEND MAP commands to a specified terminal.

During each successful SEND MAP operation, CICS stores the names of the map and mapset referenced by the SEND MAP command into the terminal control table entry for the terminal (TCTTE).

These options allow an application program to control the remembered names. This could be useful, for example, when a user invokes a help facility that sends panels using BMS facilities. In this situation, the new options allow the saving and restoring of the names of the last used application panel.

The INQUIRE and SET of map names and mapset names does not usually modify the behavior of CICS in any way. An exception to this rule is that, in a transaction routing environment in which the CICS regions are at different release levels, a SET TERMINAL command with either or both of the MAPNAME and MAPSETNAME options might cause different mapset name and map name information to be passed between the TOR and AOR.

In earlier releases, CICS stores the mapset name last used in the TCTTE and passes the mapset name in transaction routing ATTACH and DETACH requests. In this release, the map name is also passed in these requests. However, earlier releases do not recognize map names being passed in these requests.

BMS global user exits

APAR PQ12071

Documentation for PQ12071 added on 10 March 1998

There are two new global user exits for basic mapping support (BMS):

XBMIN This exit allows you to intercept a RECEIVE MAP request, after BMS has successfully processed it, if the referenced map contains at least one field that specifies VALIDN=USEREXIT, and at least one USEREXIT field is returned in the inbound data stream.

Using XBMIN, you can:

- Analyze each field defined as VALIDN=USEREXIT mapped in the current request.
- Use the mapset name, map name, field length defined in the map, and actual length of fields returned in the inbound data stream.
- Modify the data in each field.

XBMOUT This exit allows you to intercept a SEND MAP request, after BMS has successfully processed it, or, if cumulative mapping is in progress, on completion of each page of output.

Using XBMOUT, you can:

- Analyze each field defined as VALIDN=USEREXIT that has been generated in the outbound data stream.
- Use the mapset name, map name, field length defined in the map, and actual length of fields placed in the outbound data stream.
- Modify the data in each field.
- Modify the attributes sent with each field.

CICS supplies a sample global user exit program, DFH$BMXT, which shows how mapped input and output data camn be modified.

For details of the BMS XBMIN and XBMOUT global user exits, see the *CICS/ESA Customization Guide*.

Changes to the BMS map generation macros

A new operand, USEREXIT, is added for the VALIDN parameter of the DFHMDF, DFHMDI, and DFHMSD macros:

VALIDN=(USEREXIT)
Specifies that this field can be processed by the BMS global user exits, XBMIN and XBMOUT.

Prevent translator producing CBL statements

In previous releases of CICS/ESA you cannot stop the generation of a COBOL or PROCESS statement because the compiler option ALOWCBL=YES has to be specified for application programs that run under CICS.

A new CICS translator option CBLCARD|NOCBLCARD is added to enable you to control the generation of the CBL statement. CBLCARD is the default, to provide compatibility with earlier releases. NOCBLCARD stops the generation of the CBL statement.

If you specify NOCBLCARD as a translator option, you should also specify the VS COBOL II compiler option ALOWCBL=NO to avoid error message IGYOS4006-E.

Changes to the API translator options

The new translator option is described as follows:

CBLCARD|NOCBLCARD
(VS COBOL II only)

specifies whether or not the translator is to generate a CBL card. If you specify NOCBLCARD, the parameters which the CICS translator would otherwise insert must be set using VS COBOL II's IGYCOPT macro. These parameters are RENT, REUSE, NODYNAM, and LIB.

If you are using a procedure for translating, compiling, and link-editing VS COBOL II programs, you can override the translator CBLCARD default option from the job that invokes the procedure. For example, if you are using the CICS-supplied procedure, DFHEITVL, you can override the PARM parameter on the translator step as shown in the following example of the EXEC statement:

```
//     EXEC DFHEITVL,PARM.TRN=(VBREF,QUOTE,SPACE(2),NOCBLCARD)
```

Notes for installing a VS COBOL II program

To compile a VS COBOL II application program, you need the compiler options: RENT, RES, NODYNAM, and LIB. The CICS translator automatically generates a CBL statement containing these options. You can prevent the generation of a CBL or PROCESS card by selecting the translator option NOCBLCARD.

For more information about the translator option CBLCARD|NOCBLCARD, see the *CICS/ESA Application Programming Guide*. If you choose to use the NOCBLCARD option, you must also specify the VS COBOL II compiler option ALOWCBL=NO to prevent an error message of IGYOS4006-E being issued. The ALOWCBL=NO option can be overridden at compile time by the JCL PARM option or a TSO command. For more information about the ALOWCBL PARM option, see the *Application Programming Guide for MVS and CMS*. For more information about the ALOWCBL compiler option, see the *VS COBOL II Installation and Customization for MVS* manual.

CEDF enhancements

In CICS/ESA 4.1, the CICS execution diagnostic facility (EDF) provides a panel in the CEDF transaction that allows you to use CECI from the PF5 (Working Storage) key. You can use the INQUIRE and SET commands against the resources referenced by the original command before and after command execution.

The use of CECI is similar to the use of CEBR from CEDF.

Changes to externals

You can invoke the CICS CECI transaction using the PF5 key as shown in Figure 36.

PF5 : CECI
accesses CECI. This function is available from the working storage (PF5) screen. See Figure 36 for an example of the screen from which CECI is invoked. You can then use CECI commands to display or alter resource status.

```
TRANSACTION: DLID PROGRAM: DLID     TASK: 0000049 APPLID: IYAHZCIB
 ADDRESS: 00000000









WORKING STORAGE IS NOT AVAILABLE
ENTER:  CURRENT DISPLAY
PF1 : UNDEFINED           PF2 : BROWSE TEMP STORAGE PF3 : UNDEFINED
PF4 : EIB DISPLAY         PF5 : INVOKE CECI         PF6 : USER DISPLAY
PF7 : SCROLL BACK HALF    PF8 : SCROLL FORWARD HALF PF9 : UNDEFINED
PF10: SCROLL BACK FULL    PF11: SCROLL FORWARD FULL PF12: REMEMBER DISPLAY
```

Figure 36. Typical EDF display from which CECI can be invoked

+ Support for application programs written in C++

+ CICS/ESA 4.1 supports application programs written in C++. that:

+ - Are compiled using the IBM C/C++ for MVS/ESA Version 3 compiler (5655-121)
+ - Execute with the run-time libraries provided by the Language Environment for
+ MVS & VM (MVS feature) (5688-198)

+ C++ provides application programmers with all the technical and practical
+ advantages of C/370 with the benefits of object-oriented programming.

+ Changes to the application programming interface

+ You indicate to the CICS translator that your application program requires
+ translation for the C++ programming language by means of the CPP translator
+ option:

CPP
specifies that the translator is to translate C++ programs for compilation by a supported C++ compiler, such as IBM C/C++ for MVS/ESA.

Function calls are translated into the correct format depending on whether the language is C++ or C/370.

Specifying the program language to CICS

When defining program resource definitions in the CSD, you specify the language of a C++ program as LE370. CICS supports only those C++ programs that are compiled using the IBM C/C++ for MVS/ESA compiler, and which execute with run-time libraries provided by the Language Environment for MVS & VM.

Note: It is intended that future CICS support for new languages and for language enhancements will be provided only through the support available with Language Environment for MVS & VM.

Sample JCL to install C++ application programs

CICS provides two procedures to help you translate, compile and link-edit your C++ application programs. These are:

DFHYITEL The procedure for translating, compiling and link-editing CICS application programs

DFHYXTEL The procedure for translating, compiling and link-editing batch application programs that use the CICS external CICS interface (EXCI).

Figure 37 shows a sample job that invokes the DFHYITEL procedure.

```
//CPPCOMP    JOB  accounting info,name,MSGLEVEL=1
//           EXEC PROC=DFHYITEL
//TRN.SYSIN  DD      *
      #pragma XOPTS(Translator options . . .)

⋮
              C++ program source statements

⋮
/*
//LKED.SYSIN DD      *
             NAME    anyname(R)
/*
//
```

where anyname is your load module name

Figure 37. Sample job control statements to call the DFHYITEL procedure

Chapter 29. Changes for security

This chapter describes a number of changes in CICS/ESA 4.1 that affect CICS security. These items are:

- Changes to the CICS implementation of LU6.2 attach-time security
- Security facilities for the front-end programming interface (FEPI)
- Security of confidential data in CICS dumps and trace entries.

Changes to attach-time security processing and SNA profile support

CICS implementation of the LU6.2 attach-time security is changed in CICS/ESA 4.1 in conformance with the SNA architecture. As a result, you should note the following changes, which may affect your applications.

Table 40 gives the differences between the earlier releases of CICS and CICS/ESA 4.1.

Table 40. Differences in LU6.2 attach-time security processing

In earlier releases	**In CICS/ESA 4.1**
CICS ignores SNA profiles that are sent in an FMH5, and does not report errors relating to these profiles.	With the addition of profile support, CICS rejects attach requests if the FMH5 profiles are not properly coded.
CICS ignores errors or inconsistencies in the access security subfields of an FMH5.	CICS issues an error message and the attach request is rejected. The new checks are for: • An unrecognized subfield • An invalid length subfield • Multiple subfields of the same type.
+ CICS ignores the userid and password + passed in an FMH5 if they exceed 8 + characters in length, and the new + transaction is started up with default + security.	CICS accepts the full userid and password. Any trailing blanks (X'40') are removed before passing them to the external security manager, which either rejects the attach request, or can be customized to convert the userid and password into 8-character form before proceeding. **APAR PN80103** Documentation for PN80103 added on 9 May 1996
CICS accepts attach requests that do not contain security parameters in an FMH5, even if the connection specifies that security parameters are required. The new transaction is made subject to default security.	CICS rejects the attach request.
CICS accepts attach requests that have a blank, or zero length userid parameter in the attach FMH5. The new transaction is started up with default security.	CICS rejects the attach request.

APAR PN 63960

Information for APAR PN 63960 (PTF UN77613) added November 1998.

To provide some compatibility with earlier releases, a new option, USEDEFLTUSER, is added to the CONNECTION resource definition. This enables you to override some of the changes described in Table 40 on page 479. For information about attach-tiem security and specifying the USEDFLTUSER option, see the *CICS/ESA CICS-RACF Security Guide*.

Security facilities for FEPI

This section describes security enhancements for the front end programming interface (FEPI).

Overview

Because FEPI is a terminal emulator, the back-end system treats the front-end as a terminal rather than another system; it cannot differentiate between FEPI emulation and a real device. Thus, CICS bind, link, and attach-time security are not applicable to FEPI connections. If security is enabled in the back-end system, so that your FEPI application can access protected resources, the emulated terminal must be signed on to the back-end. The alternative is that you do not use CICS security with FEPI—that is, you make all the back-end transactions accessed by FEPI available to the CICS default user. This option is clearly unacceptable from a security standpoint; it means that you must either accept a security risk or deprive your FEPI applications of access to sensitive data.

In CICS/ESA 3.3, to sign-on to back-end systems, FEPI applications have either to store userids and passwords (which is risky), or to ask end-users to reenter them (which is annoying).

In CICS/ESA 4.1, FEPI applications can use the FEPI REQUEST PASSTICKET command to request that the external security manager (ESM) supplies a password substitute, known as a *PassTicket*, which can be used to sign on to the back-end system.

How to use RACF PassTickets

This section is an overview of how RACF PassTickets work, and describes what you need to do to use them. For detailed information about PassTickets, see the *RACF System Administrator's Guide*.

1. To process PassTickets, the ESM uses keys, known as secure signon keys, that are shared by the front- and back-end systems. You must define a secure signon key for each target system with which FEPI communicates. For information about how to do this, RACF users should refer to the *RACF System Programmer's Guide*. Users of other ESMs should refer to the documentation for their security product.
2. The terminal user is verified by signing on to the front-end CICS in the usual way.
3. When the user runs a transaction that uses FEPI, your application issues a FEPI REQUEST PASSTICKET command to obtain a PassTicket.

Note: The PassTicket is not displayed if you are using the CICS execution diagnostic facility (EDF).

A PassTicket is a secure representation of a password that can be used to sign on to the back-end system. It is valid for one use only, and is time-stamped. The userid for which the PassTicket is generated is that of the currently signed-on user—who can be the CICS default user (if, for example, the FEPI application is not terminal-attached). Your FEPI application can use an EXEC CICS ASSIGN command to check the userid of the currently signed-on user.

4. Your FEPI application passes the userid and PassTicket so that the back-end system can perform a signon, just as if it were sending a password and userid. For example:

```
EXEC CICS FEPI SEND FORMATTED
                   CONVID(convid) FROM(signon_data)
                   FROMLENGTH(length_of_signon_data)
```

   You must pass in the FROM data the simulated signon data, which must include the userid and the PassTicket returned on the REQUEST PASSTICKET command. It could also include the CESN transaction id to signify the type of data being passed.

   It is the application's responsibility to provide the signon processing, because CICS cannot know either the type of back-end (CICS or IMS) or the back-end program being used for signon processing.

5. The back-end system uses an unchanged interface to perform the sign-on. Thus, a CICS region receiving a userid and a PassTicket can use its existing procedures to sign on the userid. RACF takes care of the fact that a PassTicket, rather than a password, is passed to it.

Note: If the PassTicket times out (because, for example, of a session failure), your application should generate another and try to sign on again. If signon continues to fail and the front- and back-ends are in different MVS systems, check that the TOD clocks are suitably synchronized. Too many failed signon attempts could result in the userid being revoked.

For information about using RACF with CICS, see the *CICS/ESA CICS-RACF Security Guide*.

Benefits

The advantages of using PassTickets are that:

- They provide a secure way of signing on to back-end systems. This is because:
  - They are valid for one use only and are time-stamped—therefore, the potential damage caused by their being intercepted is minimal.
  - Passwords are not transmitted across the network.
- FEPI applications do not have to store passwords (or ask users to reenter them) in order to sign on to back-end systems.
- No changes are required in the CICS or IMS back-end systems.
- System clocks in the front- and back-end systems do not need to be precisely synchronized (RACF compensates for variations up to plus or minus 5 minutes).

Requirements

To use the PassTicket security mechanism for FEPI applications you need the following:

- The front-end must be a CICS/ESA 4.1 region. The back-end can be an earlier-level CICS or IMS system.
- RACF Version 2 Release 1, or a functionally-equivalent external security manager, on both the front- and back-end systems.

You must also ensure that the userid passed by the front-end to the back-end is known to the ESM called by the back-end system. In many cases the front- and back-end systems will be using the same ESM, and so this is not a problem. However, if different ESMs are involved, you must ensure that userids are defined to the ESMs at both ends.

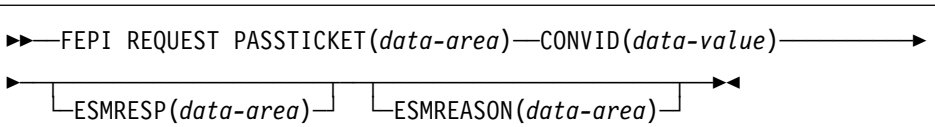
General-use programming interface

FEPI REQUEST PASSTICKET

Function

To request an external security manager (ESM) such as RACF to build a *PassTicket*. The PassTicket is a password substitute that your application can use to sign on to the back-end system associated with the conversation.

Syntax



Conditions:
INVREQ

Parameters

CONVID(8-character data-value)
specifies the ID of the conversation with the back-end system for which a PassTicket is required.

ESMREASON(fullword binary data-area)
returns the reason code from the ESM.

ESMRESP(fullword binary data-area)
returns the response code from the ESM. For an explanation of the response and reason codes returned by RACF, see the *RACF Messages and Codes* manual.

PASSTICKET(data-area)
returns the 8-byte PassTicket generated by the ESM.

Exception conditions

Condition	RESP2	Meaning
INVREQ	240	Conversation ID not owned by this task.
INVREQ	250	Passticket not built successfully.
INVREQ	251	CICS ESM interface not initialized.
INVREQ	252	Unknown return code in ESMRESP from the ESM.
INVREQ	253	Unrecognized response from CICS security modules.
INVREQ	254	Function unavailable.

End of General-use programming interface

Security of confidential data in CICS dumps and trace entries.

Confidential user data, including userids and passwords, can be found in a number of places, such as:

- VIO trace entries
- VTAM receive any input area (RAIA) in the TCP section of CICS dumps
- Traces of data received on MRO links
- Traces of FEPI request parameter list areas (RPLAREAs)
- VTAM buffer trace.

CICS/ESA 4.1 enables you to suppress the tracing of such information by means of two system initialization parameters and a new option on the transaction resource definition.

Changes to CICS externals

There are changes to CICS externals to enable you to suppress confidential data. These are:

- Changes to system definition
- Changes to resource definition.

Changes to system definition

There are two new system initialization parameters, CONFDATA and CONFTXT, to control suppression of confidential data. These are shown in Table 41.

Table 41. The DFHSIT macro parameters

	DFHSIT	[TYPE={**CSECT**|DSECT}] ⋮ [,CONFDATA={**SHOW**|HIDETC}] [,CONFTXT={**NO**|YES}] ⋮
	END	DFHSITBA

CONFDATA={SHOW|HIDETC}

Code this parameter to indicate whether CICS is to suppress (hide) user data that might otherwise appear in CICS trace entries or in dumps that contain the RAIA. This option applies to initial input data received on a VTAM RECEIVE ANY operation, the initial input data received on an MRO link, and FEPI screens and RPLAREAs.

SHOW
Data suppression is not in effect. User data is traced regardless of the CONFDATA option specified in transaction resource definitions. This option overrides the CONFDATA option in transaction resource definitions.

HIDETC
This specifies that you want CICS to 'hide' user data from CICS trace entries. It also indicates that VTAM RAIAs are to be suppressed from CICS dumps. The action actually taken by CICS is subject to the individual CONFDATA attribute on the transaction resource definition (see Table 42 on page 485).

If you specify CONFDATA=HIDETC, CICS processes VTAM, MRO, and FEPI user data as follows:

- **VTAM**: CICS clears the VTAM RAIA containing initial input as soon as it has been processed, and before the target transaction has been identified.

  The normal trace entries (FC90 and FC91) are created on completion of the RECEIVE ANY operation with the text "SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT" replacing all the user data except the first 4 bytes of normal data, or the first 8 bytes of function management headers (FMHs).

  CICS then identifies the target transaction for the data. If the transaction definition specifies CONFDATA(NO), CICS traces the user data that it suppressed from the FC90 trace in the trace entry AP FC96. This trace entry is not created if the transaction is defined with CONFDATA(YES).

- **MRO**: CICS does not trace the initial input received on an MRO link.

  The normal trace entries (DD16, DD23, and DD25) are created with the text "SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT" replacing all the user data.

  CICS then identifies the target transaction for the data. If the transaction definition specifies CONFDATA(NO), CICS traces the user data that it suppressed from DD16 in the trace entry AP FC96. This special trace entry is not created if the transaction is defined with CONFDATA(YES).

- **FEPI**: FEPI screens and RPL data areas (RPLAREAs) areas are suppressed from all FEPI trace points if CONFDATA(YES) is specified in the transaction resource definition. The user data in the FEPI trace points AP 1243, AP 1244, AP 145E, AP 145F, AP 1460, AP 1461, AP 1595, AP 1596, AP 1597, AP 1598, and AP 1599 is replaced with the message "SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT." If the transaction definition specifies CONFDATA(NO), the FEPI trace entries are created with the user data as normal.

Mirror transactions: The CICS-supplied mirror transaction definitions are specified with CONFDATA(YES). This ensures that, when you specify CONFDATA=HIDETC as a system initialization parameter, CICS regions running mirror transactions suppress user data as described for VTAM and MRO data.

Modified data: By waiting until the transaction has been identified to determine the CONFDATA option, VTAM or MRO data may have been modified (for example, it may have been translated to upper case).

The interaction between the CONFDATA system initialization parameter and the CONFDATA attribute on the transaction resource definition is shown in Table 42.

Table 42. Effect of CONFDATA system initialization and transaction definition parameters

CONFDATA on transaction	CONFDATA system initialization parameter	
	SHOW	HIDETC
NO	Data not suppressed	VTAM RAIAs are cleared. Initial input of VTAM and MRO data is suppressed from the normal FC90, FC91, DD16, DD23, and DD25 trace entries. For FC90 and DD16 traces only, suppressed user data is traced separately in an FC96 trace entry. FEPI screens and RPLAREAs are traced as normal.
YES	Data not suppressed	VTAM RAIAs are cleared. All VTAM, MRO, and FEPI user data is suppressed from trace entries.

You cannot modify the CONFDATA option while CICS is running. You must restart CICS to make such a change.

Restrictions

You can code the CONFDATA parameter in the SIT, PARM, and SYSIN only.

CONFTXT={NO|YES}

Code this parameter to indicate whether CICS is to prevent VTAM from tracing user data.

NO
CICS does not prevent VTAM from tracing user data.

YES
CICS prevents VTAM from tracing user data.

You cannot modify the CONFTXT option while CICS is running. You must restart CICS to make such a change.

Restrictions

You can code the CONFTXT parameter in the SIT, PARM, and SYSIN only.

Changes to resource definition

The CONFDATA option is added to the transaction resource definition, as shown in Figure 38 on page 486.

```
 TRansaction  ==>
 Group        ==>
 .
 .
RECOVERY
 DTimout      ==> No               No | 1-6800
 INdoubt      ==> Backout          Backout | Commit | Wait
 RESTart      ==> No               No | Yes
 SPurge       ==> No               No | Yes
 TPUrge       ==> No               No | Yes
 DUmp         ==> Yes              Yes | No
 TRACe        ==> Yes              Yes | No
 COnfdata     ==> No               No | Yes
SECURITY
 RESSec       ==> Yes              No | Yes
  .
  .
  .
```

Figure 38. The DEFINE panel for TRANSACTION

CONFDATA(NO|YES)

Indicates whether CICS is to suppress user data from CICS trace entries when the CONFDATA system initialization parameter specifies HIDETC. If the system initialization parameter specifies CONFDATA=SHOW, CONFDATA on the transaction definition is ignored.

If the system initialization parameter specifies CONFDATA=HIDETC, the following options are effective:

NO

CICS does not suppress any user data. VTAM and MRO initial user data is traced in trace point AP FC96. FEPI user data is traced in the normal CICS FEPI trace points.

YES

CICS suppresses user data from the CICS trace points.

Chapter 30. Changes for sysplex exploitation

This chapter describes a number of changes introduced in CICS/ESA 4.1 to support the MVS sysplex environment. These items are:

- CICS use of VTAM generic resources for all terminal-owning regions in a CICSplex
- Elimination of need for transaction definitions in a terminal-owning region.

+ In a sysplex, you can exploit some of the benefits of shared data tables support for
+ data that is primarily read-only. You can do this by replicating user-maintained data
+ tables in each MVS image, so that the data can be accessed by CICS regions
+ across the sysplex. For more information, see the *CICS/ESA Shared Data Tables*
+ *Guide*.

VTAM generic resource registration

This section describes how CICS/ESA 4.1 uses the VTAM generic resources function. It covers the following topics:

- Overview
- Benefits
- Requirements
- Changes to CICS externals
- Changes to security.

Overview of VTAM generic resources function

A VTAM application program such as CICS can now be known by its generic resource name, in addition to its own application program network name as defined on the APPL definition statement. A number of CICS regions can use the same generic resource name. VTAM keeps a map of the CICS APPLIDs that are members of a generic resource name.

Each CICS region that uses a given generic resource name is a member of that generic resource name set.

A terminal user, wishing to start a session with a CICSplex that has several terminal-owning regions, uses the generic resource name in the logon request. Using the generic resource name, VTAM establishes the session with one of the member CICS regions that is registered as a member of the generic resource name.

For the generic resource function to operate, each VTAM application must register itself to VTAM under its generic resource name. Registration is performed automatically by CICS when it is ready to receive VTAM logon requests.

CICS use of the VTAM generic resource function

CICS uses the VTAM generic resource function by means of the following additions to CICS/ESA 4.1:

- Support for registration, and corresponding de-registration, as a member of a generic resource name

- A new system initialization parameter to specify the generic resource name
- A new CICS INQUIRE command to obtain the generic resource name.

Registration and de-registration: CICS registers its APPLID under a specified generic resource name at the following times:

- During CICS initialization
- Whenever the VTAM ACB is reopened, after being closed for any reason while CICS is running.

CICS de-registers from a generic resource whenever the VTAM ACB is closed. This could occur as a result of:

- A command to close the VTAM ACB, either an EXEC CICS SET VTAM CLOSED or a CEMT SET VTAM CLOSED command
- A command to shut down CICS, either an EXEC CICS PERFORM SHUTDOWN or a CEMT PERFORM SHUTDOWN command.

The new system initialization parameter and the INQUIRE commands are described under "Changes to CICS externals" on page 489.

Restrictions for certain device types

There are some restrictions on the use of generic resources by certain device types:

- Devices using message protection cannot logon using the generic resource name
- There are restrictions that affect LUTYPE6 connections with CICS regions that are members of a VTAM generic resource name. These are described in the *CICS/ESA Intercommunication Guide*.

For information about defining RACF security profiles for partners on APPC connections, where the CICS partners are members of a generic resource name, see the *CICS/ESA CICS-RACF Security Guide*.

Benefits

The main benefits accrue when VTAM generic resource registration is applied to CICS terminal-owning regions. This enables VTAM to perform dynamic workload balancing of the terminal sessions across the available terminal-owning regions.

The terminal-owning regions can in turn perform dynamic workload balancing using the CICS dynamic transaction routing facility, which leads to improved performance overall and faster turnaround for individual transactions.

Requirements

To use VTAM generic resources in CICS/ESA 4.1, you need:

- ACF/VTAM Version 4 Release 2 or a later, upward-compatible, release.

In addition, VTAM 4.2 must be:

- Running under an MVS that is part of a sysplex
- Connected to an MVS coupling facility.

For information about the sysplex coupling facility, see the *MVS/ESA Setting Up a Sysplex* manual, GC28-1449.

Changes to CICS externals

There are additions to the following CICS externals to enable you to define and use a VTAM generic resource name with CICS:

- Changes to system definition
- Changes to the system programming interface
- Changes to CICS-supplied transactions.

Changes to system definition

A new system initialization parameter is added to enable you to define the generic resource name under which CICS is to register. The parameter is shown in Table 43.

Table 43. The DFHSIT macro parameters

	DFHSIT	TYPE={**CSECT**|DSECT} ... [,GRNAME=name] ...
	END	DFHSITBA

GRNAME=name

Specifies the VTAM generic resource name under which a group of CICS terminal-owning regions in a CICSplex register to VTAM.

There is no default for GRNAME. If you don't specify GRNAME, CICS does not register itself with the VTAM generic resources function.

Notes:

1. If you are operating a CICSplex that comprises separate terminal-owning regions and application-owning regions, you should ensure that you define a VTAM generic resource name to the CICS terminal-owning regions only.
2. The generic resource name can be 1 through 8 characters, but you are recommended to specify 8 characters, using a generic symbol to replace trailing blanks as shown in the following example:

   ```
   GRNAME=CICSD###
   ```

3. The GRNAME and XRF system initialization parameters are mutually exclusive. If you specify XRF=YES, you should not specify a value for the GRNAME system initialization parameter. If you specify XRF=YES, any value specified for GRNAME is set to blanks.
4. If you specify a valid generic resources name, you should specify *name1* only on the APPLID system initialization parameter. If you specify *name1* and *name2* on the APPLID parameter, CICS ignores *name1* and uses *name2* as the VTAM applid.

The examples used here are based on a CICS naming convention described in the *System/390 MVS Sysplex Application Migration* manual, GC28-1211.

Changes to the system programming interface

General-use programming interface

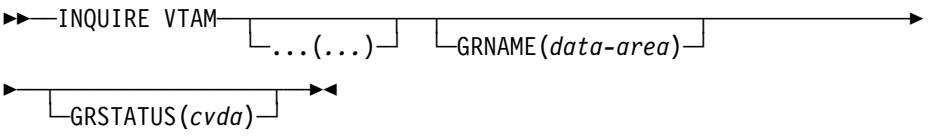
The INQUIRE VTAM command is extended to enable you to obtain the VTAM generic resource name under which a CICS terminal-owning region is registered.

INQUIRE VTAM

Function

Inquire on the state of the connection between CICS and VTAM and also obtain the VTAM generic resource name.

Syntax



Conditions:
INVREQ, NOTAUTH

INQUIRE VTAM options

GRNAME(data-value)
returns the 8-byte generic resource name under which this CICS region is registered.

GRSTATUS(data-area)
returns a CVDA value indicating the status of generic resource registration. CVDA values are:

DEREGERROR
CICS has unsuccessfully attempted to de-register as a member of the VTAM generic resources name specified by the GRNAME system initialization parameter.

DEREGISTERED
CICS has successfully de-registered as a member of the generic resources name specified by the GRNAME system initialization parameter.

NOTAPPLIC GRNAME is not specified as a system initialization parameter for this CICS region.

REGERROR CICS has unsuccessfully attempted to register as a member of the generic resource name specified by the GRNAME system initialization parameter.

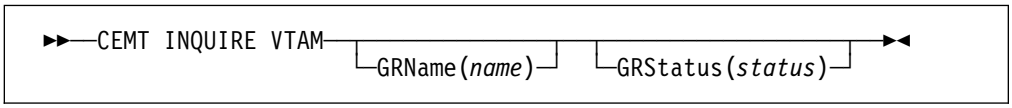
REGISTERED
CICS registered successfully as a member of the VTAM generic resource named by the GRNAME system initialization parameter.

UNAVAILABLE
A generic resource name (GRNAME) was specified, but the version of VTAM that CICS is connected to does not support generic resources.

UNREGISTERED
A generic resource name (GRNAME) was specified, but CICS has not yet made any attempt to register.

Changes to CICS-supplied transactions

The CEMT INQUIRE VTAM command now includes the GRNAME option to enable you to display the generic resource name under which this CICS region is registered to VTAM.



GRName(name)
displays the generic resource name under which this CICS region is registered.

GRStatus(status)
displays the status of generic resource registration. The status values are:

REGistered CICS registered successfully as a member of the VTAM generic resource named by the GRNAME option.

DERegistered
CICS was registered as a member of the generic resource named by the GRNAME and has de-registered successfully.

UNRegistered
A generic resource name (GRNAME) was specified, but CICS has not made any attempt to register.

UNAvailable A generic resource name (GRNAME) was specified, but the version of VTAM that CICS is connected to does not support generic resources.

If CICS makes an unsuccessful attempt to register, the status is shown as UNREGISTERED, and the words REGISTER ERROR appear alongside.

If CICS makes as unsuccessful attempt to de-register, the status is shown as REGISTERED, and the words DEREGISTER ERROR appear alongside.

End of General-use programming interface

+ Changes to security

+ There is a new security requirement if you are using VTAM generic resources, to
+ enable your CICS regions to register under their generic resource name.

Authorizing access to VTAM generic resources

If your CICS terminal-owning regions (TORs) specify a VTAM generic resource name (on the GRNAME system initialization parameter), the TORs must first be authorized to register under the specified generic resource name.

To authorize CICS terminal-owning regions to access a VTAM generic resource, you must define a VTAMAPPL profile with the generic resource name as the VTAMAPPL profile name. Authorize each CICS terminal-owning region with READ access to the VTAMAPPL profile.

For example, if the generic resource name CICSD### is used by 4 TORs under the CICS region userids of CICSDTA1, CICSDTB1, CICSDTC1, and CICSDTD1, you can define the profile and authorizations as follows:

```
RDEFINE  VTAMAPPL  CICSD###  UACC(NONE)

PERMIT CICSD### CLASS(VTAMAPPL) ID(CICSDTA1, CICSDTB1, CICSDTC1, CICSDTD1)
          ACCESS(READ)
```

Elimination of need for transaction definitions in a terminal-owning region

This section describes how CICS/ESA 4.1 eliminates the need for resource definitions in a terminal-owning region for transactions that are to be dynamically routed. It covers the following topics:

- Overview
- Benefits
- Changes to CICS externals.

Overview

CICS/ESA Version 3 permits the use of a single transaction resource definition to serve as both remote and local definitions in terminal-owning and application-owning regions respectively. These are referred to as dual-purpose resource definitions.

This method works efficiently when the transaction is defined for static transaction routing, with a REMOTESYSTEM name and DYNAMIC(NO). However, if the transaction is defined with DYNAMIC(YES), CICS invokes a user-written dynamic transaction routing program to determine the target application-owning region. This invocation of the dynamic transaction routing program is undesirable in an application-owning region, when the transaction has already been routed by a terminal-owning region, because of the unnecessary overhead incurred in the application-owning region. The alternative in CICS/ESA Version 3 is to maintain two sets of transaction definitions.

CICS/ESA 4.1 avoids the extra administration effort involved in maintaining two sets of resource definitions, by eliminating the need for a transaction resource definition in the terminal-owning region when the transaction is to be dynamically routed.

New system initialization parameter

A new system initialization parameter, DTRTRAN, is introduced to specify a dynamic transaction routing resource definition. This provides a CICS region with a set of default attributes for those transactions that are to be dynamically routed. It is intended primarily for use in terminal-owning regions, but can also be used in application-owning regions where a transaction is being routed across several regions ("daisy-chained").

New COMMAREA parameters for the dynamic routing exit

Two new parameters are added to the COMMAREA passed to the dynamic transaction routing program to support the common transaction definition specified on the DTRTRAN system initialization parameter. These are:

```
DYRDTRXN DS    CL1                    DTRTRAN INDICATOR (Y/N)
DYRDTRRJ DS    CL1                    DTRTRAN REJECT (Y/N)
```

For details of these parameters, see Chapter 6, "Dynamic transaction routing enhancements" on page 99.

How the new method works

Under the new method, you need define only one transaction definition for use in a terminal-owning region for all dynamically routed transactions.

At transaction attach, CICS uses the services of directory manager to find an installed resource definition for the user transaction identifier (transid). If there is no definition for the user transid, and the transaction is a candidate for dynamic routing, CICS checks to see if there is an installed definition for the transaction specified on the DTRTRAN system initialization parameter, and processes the transaction as follows:

- If the transaction resource definition specified on the DTRTRAN parameter is ***not*** installed, CICS attaches the CICS-supplied transaction CSAC. This returns DFHAC2001—the transaction '*tranid*' is unrecognized message—to the user terminal.
- If the transaction resource definition specified on the DTRTRAN parameter ***is*** installed, CICS attaches a transaction built from the following information:
  - The user transaction identifier, and
  - The set of attributes taken from the transaction definition specified on the DTRTRAN parameter.

  Because the dynamic transaction routing transaction is specified with DYNAMIC(YES), CICS invokes the dynamic transaction routing program to select a target application-owning region and, if necessary, name the remote transaction.

The processing in an application-owning region is as follows:

- If a transaction is routed to an application-owning region which has a resource definition installed for it, the transaction is executed as usual.
- If a transaction is routed to an application-owning region which does not have a definition installed for it, the action CICS takes depends on the existence of a dynamic transaction routing resource definition, as follows:

- If the transaction resource definition specified on the DTRTRAN parameter is installed, CICS invokes the dynamic transaction routing program to determine which region it should next be routed ("daisy-chained").
- If the transaction resource definition specified on the DTRTRAN parameter is ***not*** installed, CICS attaches the CICS-supplied transaction CSAC. This returns DFHAC2001—the transaction '*tranid* ' is unrecognized message—to the user terminal.

  Unless you specifically want to daisy-chain transaction routing requests, you are recommended not to install a DTRTRAN in an application-owning region.

You can also continue to define and install individual transaction resource definitions in terminal-owning regions, and CICS continues to use these as before.

Transactions invoked by special task request function

If you have applications in which terminal users invoke transactions by special task request functions, as defined by the TASKREQ attribute of the transaction resource definition, you must define specific transaction resource definitions for them.

The elimination of the need for transaction definitions applies only to those transactions that are invoked by a tranid.

How to use the new dynamic transaction routing definition

As an illustration of how you would use the new dynamic transaction routing definition in a dynamic transaction routing environment, the following summarizes the transaction definitions that you would typically install in the terminal-owning regions and application-owning regions:

- In the terminal-owning region:
  - The CICS-supplied transactions.
  - The dynamic transaction routing resource definition specified on the DTRTRAN system initialization parameter.
  - The user transactions that are to run in the terminal-owning region. For example, sign-on and menu transactions.
- In the application-owning region:
  - The CICS-supplied transactions.
  - The user transactions that run in the application-owning region; that is, all the transactions that are routed to the application-owning region by one or more terminal-owning regions.
  - The remote transactions that are to be started by an EXEC CICS START command.

    If a started transaction specifies a terminal, the schedule request for the terminal may be shipped to a terminal-owning region for subsequent transaction routing to the region where it is to execute. In this case, you do not need a specific transaction definition in the terminal-owning region.

In general, the ability to dynamically route transactions without an explicit transaction definition should enable you to install a transaction resource definition for a given transaction in either the application-owning region or the terminal-owning region.

However, you may have some transactions that run either in the local region, or in a remote region, depending on input data. In cases like this, the transaction must be defined in both regions, with the definition in the terminal-owning region defined as dynamic.

Transactions that are started by an EXEC CICS START command to run on a remote system also require remote definitions.

This greatly simplifies the task of managing resource definitions. However, you can continue to use existing remote and dynamic definitions if more complicated configurations are required, or as you migrate from the old method to the new.

Benefits

Eliminating the need for transaction definitions in the terminal-owning region offers one of the following benefits:

- A significant saving in system administration effort, because you don't need to maintain two resource definitions for the dynamic transaction routing environment.
- An improvement in performance because you don't need to incur the unnecessary overhead of the dynamic transaction routing program in the application-owning region.

Changes to CICS externals

The elimination of the transaction resource definition from the terminal-owning region affects the following CICS externals:

- Changes to system definition
- Changes to CICS-supplied transactions
- Changes to system programming interface.

Changes to system definition

A new system initialization parameter is added to enable you to define the dynamic transaction routing definition that you want CICS to use for dynamic transaction routing. The parameter is shown in Table 44.

Table 44. The DFHSIT macro parameters

	DFHSIT	TYPE={**CSECT**|DSECT} ...
+		[,DTRTRAN={**CRTX**|name|NO} ...
	END	DFHSITBA

DTRTRAN={CRTX|name}

This is the name of the transaction definition that you want CICS to use for dynamic transaction routing. This is intended primarily for use in a CICS terminal-owning region, although you can also use it in an application-owning region when you want to daisy-chain transaction routing requests. In a dynamic transaction routing environment, the transaction named on DTRTRAN must be installed in the CICS terminal-owning regions if you want to eliminate the need for resource definitions for individual transactions.

See the description of the new CRTX transaction on page 496 for information about defining a dynamic transaction routing definition.

The transaction name is stored in the catalog for recovery during CICS restarts.

CRTX
This is the default dynamic transaction definition. It is the name of the CICS-supplied sample transaction resource definition provided in the CSD group DFHISC.

name
The name of your own dynamic transaction resource definition that you want CICS to use for dynamic transaction routing.

+ **NO**
+ Do not invoke dynamic transaction routing automatically if a transaction
+ resource definition cannot be found.

Changes to CICS-supplied transactions.

There is a new CICS-supplied transaction, CRTX, provided to support dynamic transaction routing.

The CRTX transaction: The CICS-supplied dynamic transaction routing transaction definition, CRTX, is supplied in the CSD group, DFHISC, and this group is included in the CICS-supplied startup group list, DFHLIST.

The main attributes of the CRTX transaction definition are shown in Figure 39.

```
OBJECT CHARACTERISTICS                                  CICS RELEASE = 0410
 CEDA Define
  TRansaction    : CRTX
   ...
  PROGram        : ########
  TWasize        : 00000             0-32767
  PROFile        : DFHCICST
   ...
  STatus         : Enabled           Enabled | Disabled
   ...
  TASKDATALoc    : Any               Below | Any
  TASKDATAKey    : CICS              User | Cics
 REMOTE ATTRIBUTES
  DYnamic        : YES               No | Yes
  REMOTESystem   :
  REMOTEName     :
  TRProf         : DFHCICSS
   ...
 RECOVERY
  DTimout        : No                No | 1-6800
  Indoubt        : Backout           Backout | Commit | Wait
  RESTart        : No                No | Yes
  SPurge         : YES               No | Yes
  TPUrge         : YES               No | Yes
   ...

                                                        APPLID=CICSHTA1

PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 39. Main attributes of the CICS-supplied CRTX transaction

- If you do not specify your transaction id on the DTRTRAN system initialization parameter, CICS uses CRTX by default.
- You are recommended to create your own DTRTRAN transaction, using CRTX as a model.
- You must specify DYNAMIC(YES) in the DTRTRAN transaction definition.

The key parameters of this transaction definition are described below:

DYNAMIC(YES)
This is required for a dynamic transaction routing definition that is specified on the DTRTRAN system initialization parameter. You can change the other parameters when creating your own definition, but must specify DYNAMIC(YES).

PROGRAM(########)
The CICS-supplied default transaction specifies a dummy program name, ########. If your dynamic transaction routing program allows a transaction to run in the local region, and its definition specifies the dummy program name, CICS is unlikely to find such a program, causing a "program-not-found" condition.

You are recommended to specify the name of a program that you want CICS to invoke whenever the transaction:

- Is not routed to a remote system, and
- Is not rejected by the dynamic transaction routing program by means of the DYRDTRRJ parameter, and
- Is run in the local region

You can use the local program to issue a suitable response to a user's terminal in the event that the dynamic routing program decides it cannot route the transaction to a remote system.

TRANSACTION(CRTX)
The name of the CICS-supplied dynamic transaction routing definition. Change this to specify your own transaction identifier.

RESTART(NO)
This attribute is forced for a routed transaction.

Changes to the system programming interface

General-use programming interface

The use of a dynamic transaction routing definition in a terminal-owning region can affect the results of an EXEC CICS INQUIRE TRANSACTION command:

- If you specify the user transaction identifier on an EXEC CICS INQUIRE TRANSACTION command, CICS returns TRANSIDERR because there is no entry in the table of installed transaction resource definitions.

However, if you inquire on a user task that was attached using the dynamic transaction routing definition, the EXEC CICS INQUIRE TASK command returns the user transaction identifier—not the DTRTRAN name.

End of General-use programming interface

Changes to monitoring and statistics

CICS monitoring data contains the user transaction identifier.

However, all transaction-related statistics are accrued against the transaction identifier specified on the DTRTRAN parameter.

Chapter 31. Changes for system and resource definition

This chapter describes a number of changes in CICS/ESA 4.1 that affect system and resource definition. These items are:

- Change to generation of session names (TCTTE entries)
- Indirect links for transaction routing
- Allow multiple group lists on CICS startup
- Addition of NETNAME to the DFHZATD parameter list
- Addition of a new CICS-supplied PROFILE definition, DFHCICSP
- Addition of a new CSD compatibility group, DFHCOMP4
- Common destination control tables for a CICSplex
- New system initialization parameter to specify CICS use of LLACOPY or BLDL
- Increased CI size and number of buffers for temporary storage
- Extra buffers added for transient data
- Efficient deletion of shipped terminal definitions
- Extensions to the DFHTST macro for easier definition of local and remote TS queues.

Change to generation of session names (TCTTE entries)

The generation of session names is changed for both MRO and APPC sessions.

Generation of MRO session names

In CICS/ESA 4.1 you can let the send and receive prefix parameters default when defining SESSIONS resource definitions for MRO connections.

Alternatively, you can continue to define your own prefixes for the send and receive sessions, in which case CICS generates the terminal control table terminal entries (TCTTEs) for session names in the same way as in earlier releases for LU6.1 ISC sessions and MRO sessions. (That is, CICS creates session names by using the 1- or 2-character prefix followed by a CICS-generated numeric string in range 1-999.)

Defining SESSIONS for MRO with default prefixes

In CICS/ESA 4.1, CICS sets default values for the RECEIVEPFX and SENDPFX options when you are defining a SESSIONS resource definition and the PROTOCOL parameter specifies LU61.

If you omit the prefix parameters from the command, or leave them blank on the CEDA define panel, CICS sets a default prefix of a greater-than symbol (>) for send sessions, and a less-than symbol (<) for receive sessions. CICS uses the prefix in conjunction with the send and receive counts to generate the session names, using the following method:

Unique MRO prefixes

CICS generates the names for send and receive sessions using the following default prefixes:

- A greater-than symbol (>) as a unique prefix for send sessions
- A less-than symbol (<) as a unique prefix for receive sessions.

A 3-character alpha-numeric value

CICS creates the last three characters of the session names from the alphanumeric characters A through Z, and 1 through 9. The generated 3-character identifiers begin with the letters AAA, and continue in ascending sequence until the number of session entries reaches the limit set by the SENDCOUNT and RECEIVECOUNT values.

Note that CICS starts the alphanumeric sequence when installing the first send session, continues through its partner receive sessions, and then continues the sequence with the next connection it installs.

For example, if you install the first MRO connection with an associated sessions definition that specifies a send and receive count of 10, CICS generates the sessions names as follows:

Send session names allocated are: >AAA through >AAJ
Receive session names allocated are: <AAK through <AAT

If you install a second connection, the send session names for that connection begin with >AAU and continue in ascending sequence.

For LU61 sessions, you should continue to specify your own prefix from the permitted range of characters, but omitting the > and < symbols.

Generation of APPC session names

In earlier releases, CICS creates session names by appending a 3-character alphanumeric string to the – (minus) symbol. Names begin with 999 and continue in *descending* sequence.

In CICS/ESA 4.1 this is changed to ascending sequence, beginning with –AAA, as for the MRO sessions described above.

Indirect links for transaction routing

This section describes changes to the need to define indirect links between CICS systems.

In a CICS/ESA 3.3 transaction routing environment, where the terminal-owning region (TOR) and application-owning region (AOR) are not directly connected, you must define an indirect connection to the TOR in the AOR and in the intermediate systems. Thus, in CICS/ESA 3.3, an indirect link has to be defined in every system in a transaction routing path from a TOR to an AOR, except for the TOR itself and the first region in the path (that is, the region to which the TOR has a direct link).

The purpose of indirect links was to allow CICS to fully identify remote terminals (by providing the netname of the TOR) and to point to the next link in the path to the TOR. They were needed to allow:

- Terminal definitions to be shipped to and installed in the AOR
- Transactions routed from the TOR to be attached
- ATI-initiated transaction routing (that is, where an ATI request on the AOR is associated with a terminal owned by a remote TOR)
- A transaction running on the AOR to acquire an alternative facility to an APPC device (by issuing an ALLOCATE command).

For further information about indirect links, and how to define them, see the *CICS/ESA Intercommunication Guide*.

In CICS/ESA 4.1, it is only necessary to define indirect links if you use non-VTAM terminals for transaction routing across intermediate systems. Optionally, you can use them with VTAM terminals, where several transaction routing paths are possible, via different sets of intermediate systems, to identify the preferred path to the TOR. Even for this purpose, they are not required if you are using statically-defined terminals—see below.

You can also define indirect links on some intermediate systems on the transaction routing path, and not on others.

Shippable terminals

Indirect links are no longer necessary to allow terminal definitions to be shipped to an AOR across intermediate systems. Each shipped definition contains a pointer to the previous system in the transaction routing path (or to an indirect connection to the TOR, if one exists). This allows routed transactions to be attached, by identifying the netname of the TOR and the path from the AOR to the TOR.

If several paths are available, you can use indirect links to specify the preferred path to the TOR.

Note: Non-VTAM terminals are not shippable.

Statically-defined terminals

There may be times when you cannot use, or choose not to use, shippable terminals. For example, you may have transactions that issue ATI requests associated with a remote terminal, or ALLOCATE commands associated with a remote APPC device, where you cannot rely on a terminal definition having been shipped to the AOR by a previous transaction routing operation. (The terminal may be an output-only device, such as a printer.) If you do not or cannot use shipped definitions, you must statically define remote terminals to the AOR and the intermediate systems.

If you are using VTAM terminals, indirect links are no longer required. You use the new REMOTESYSNET option of the TERMINAL definition (or the CONNECTION definition, if the “terminal” is an APPC device) to specify the netname of the TOR; and the REMOTESYSTEM option to specify the next system in the path to the TOR. If several paths are available, use REMOTESYSTEM to specify the next system in the preferred path.

If you are using non-VTAM terminals, indirect links are required. This is because you cannot use RDO to define non-VTAM terminals; the DFHTCT TYPE=REMOTE or TYPE=REGION macros used to create the remote definitions do not include an equivalent of the REMOTESYSNET option of CEDA DEFINE TERMINAL.

Changes to resource definition

A new option, REMOTESYSNET, is added to the CEDA DEFINE TERMINAL and DEFINE CONNECTION commands. Also, the meaning of the REMOTESYSTEM option on the same commands has changed.

TERMINAL definition

REMOTESYSNET(name)

The network name (APPLID) of the region that owns the terminal. The name can be up to eight characters in length. It follows assembler language rules, and must start with an alphabetic character. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase.

REMOTESYSNET is used where there is no direct link between the region in which this definition is installed and the terminal-owning region. You do not need to specify REMOTESYSNET if either of the following is true:

- You are defining a local terminal (that is, REMOTESYSTEM is not specified, or specifies the name of the local system).
- REMOTESYSTEM names a direct link to the terminal-owning region. However, there is one special case: if the terminal-owning region is a member of a VTAM generic resources group and the direct link is an APPC connection, you **do** need to specify REMOTESYSNET.

REMOTESYSTEM(name)

The name that identifies the intercommunication link to the system that owns the terminal. The name can be up to four characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase.

The remote system name should be the CONNECTION name on the CONNECTION definition for the intercommunication link. If it is not specified, or if it is specified as the name of the local system, then this terminal is local to the region in which this resource definition is installed. If the name is that of another region, the terminal is remote. You can therefore use the same resource definition for the terminal in both the local system and a remote system.

If there are intermediate systems between this CICS and the terminal-owning region, REMOTESYSTEM should specify the first link in the path to the TOR. If there is more than one possible path, it should specify the first link in the preferred path.

REMOTESYSTEM is ignored if you specify AUTINSTMODEL(YES) or (ONLY).

CONNECTION definition

REMOTESYSNET(name)

The network name (APPLID) of the system that owns the connection. The name can be up to eight characters in length. It follows assembler language rules, and must start with an alphabetic character. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase.

Use REMOTESYSNET when transaction routing to remote APPC systems or devices and there is no direct link between the region in which this definition is installed and the system that owns the connection to the remote device. You do not need to specify REMOTESYSNET if either of the following is true:

- You are defining a local connection (that is, REMOTESYSTEM is not specified, or specifies the name of the local system).
- REMOTESYSTEM names a direct link to the system that owns the connection. However, there is one special case: if the connection-owning

region is a member of a VTAM generic resources group and the direct link to it is an APPC connection, you **do** need to specify REMOTESYSNET.

REMOTESYSTEM(name)
The name that identifies the intercommunication link to the system that owns the connection. The name can be up to four characters in length. The acceptable characters are: A-Z 0-9 $ @ and #. Lowercase characters are converted to uppercase.

This is the CONNECTION name on the CONNECTION definition for the intercommunication link.

REMOTESYSTEM is used for transaction routing to remote APPC systems or devices. If it is not specified, or if it is specified as the name of the local system, then this connection will be local to this system. If the name is that of another system, the connection will be remote. You can therefore use the same definition for the connection in both the local system and a remote system.

If there are intermediate systems between this CICS and the region that owns the (connection to) the device, REMOTESYSTEM should specify the first link in the path to the device-owning region. If there is more than one possible path, it should specify the first link in the preferred path.

Allow multiple group lists on CICS startup

In CICS/ESA 4.1, you can specify up to four resource definition group list names on the GRPLIST system initialization parameter. Each name can be either a real group list name or a generic group list name that incorporates generic symbols (+ and *). If you specify more than one group list (either by specifically coding two or more group list names or by coding a group list name with generic symbols), the later group lists are concatenated onto the first group list. Any duplicate resource definitions in later group lists override those in earlier group lists.

The changes to the GRPLIST system initialization parameter are shown in Table 45. For details of all the system initialization parameters available in CICS/ESA 4.1 see the *CICS/ESA System Definition Guide*.

Table 45. GRPLIST system initialization parameter syntax

		[,GRPLIST={**DFHLIST**|name| (name[,name2][,name3][,name4])}]

GRPLIST={DFHLIST|name|(name[,name2][,name3][,name4])}
Specifies the names (each 1 through 8 characters) of up to four lists of resource definition groups on the CICS system definition (CSD) file. The resource definitions in all the groups in the specified lists are loaded during initialization when CICS performs a cold start. If CICS performs a warm or emergency start, the resource definitions are restored from the global catalog, and the GRPLIST parameter is ignored.

Each name can be either a real group list name or a generic group list name that incorporates the generic characters + or *. If you specify more than one group list (either by specifically coding two or more group list names or by coding a generic group list name) the later group lists are concatenated onto

the first group list. Any duplicate system resource definitions in later group lists override those in earlier group lists.

Using generic names enables you to specify more than four lists. CICS loads groups from any list that matches a generic name, where the + symbol represents any single non-blank character, and the * represents any number of non-blank characters. CICS processes lists that match a generic name in EBCDIC alphanumeric collating sequence.

Use the CEDA command LOCK to protect the lists of resource groups specified on the GRPLIST parameter.

The default is DFHLIST, the CICS-supplied list that specifies the set of resource definitions needed by CICS. Do not code GRPLIST=NO unless you have a group list named NO.

For more information about the GRPLIST parameter, see the *CICS/ESA System Definition Guide*. For more information about resource definitions, groups, group lists, and the CSD, see the *CICS/ESA System Definition Guide*.

To override one or more of the group lists specified on the GRPLIST system initialization parameter, you must specify all list names that you want to use, even if you are not changing the names. For example, if your DFHSIT macro contains the parameter:

```
GRPLIST=(GRPLST01,GRPLST02,GRPLST03,GRPLST04)
```

and you want to replace the list GRPLST03 with the list ANOLST05, you should specify the override:

```
GRPLIST=(GRPLST01,GRPLST02,ANOLST05,GRPLST04)
```

In general, any required resource definitions should appear in *one of* the group lists specified on the GRPLIST system initialization parameter.

Addition of NETNAME to the DFHZATD parameter list

In CICS/ESA 4.1, the output parameter list set up by the autoinstall control program DFHZATD contains the NETNAME of the autoinstalled terminal whose TCTTE has been deleted.

This is of benefit as networks become larger and more complex, and tighter control is needed.

Table 46 shows the contents of the parameter list at DELETE time.

Table 46. Autoinstall control program's parameter list at DELETE

	1st byte	**2nd byte**	**3rd byte**	**4th byte**
First fullword	X'F1'	'Z'	'C'	Reserved
Second fullword	ID of terminal to be deleted			
Third fullword	Length of netname to be deleted		First two bytes of netname	
Next 15 bytes	Remainder of netname			

Addition of a new CICS-supplied PROFILE definition, DFHCICSP

CICS provides a new PROFILE definition, DFHCICSP, in the CSD group DFHSTAND. The new profile is identical to DFHCICST except that it specifies UCTRAN(YES) instead of UCTRAN(NO).

The new profile is used by the CICS-supplied page retrieval transaction, CSPG. The new profile, together with changes in the task-attach routine and the page retrieval program, enables CICS to perform upper case translation at the transaction level for BMS paging.

This allows users of terminals that are defined with uppercase translation switched off to use the page retrieval function without having to enter paging commands in upper case. Assigning a new profile for CSPG means that all data entered on the retrieval command (defined by the system initialization parameter PGRET) and the purge command (defined by the system initialization parameter PGPURGE) is translated to uppercase.

If a user's terminal is defined with UCTRAN(YES), the new profile has no effect since all terminal input is translated to uppercase anyway.

See the *CICS/ESA CICS-Supplied Transactions* manual for information about the CSPG transaction.

Addition of a new CSD compatibility group, DFHCOMP4

Some CICS-supplied resource definitions are changed or are obsolete in CICS/ESA 4.1, and are moved to the compatibility group, DFHCOMP4. This group is required for compatibility between CICS/ESA 4.1 and CICS/ESA 3.3. It contains the following programs:

DFHCRP
DFHCSSC
DFHNEP
DFHRTY
DFHSNT

and the following transactions:

CEDF
CLS1
CLS3
CSPG
CSSC

You must add DFHCOMP4 to any CICS/ESA 3.3 group list if you plan to share the CSD between CICS/ESA 3.3 and CICS/ESA 4.1.

See the *CICS/ESA Migration Guide* for more information about DFHCOMP4 and other compatibility groups.

Common destination control tables for a CICSplex

Changes are made in CICS/ESA 4.1 to the macro-level resource definitions for the destination control table (DCT).

In earlier releases, a transient data (TD) queue can be defined either as a local queue, or a remote queue, but not both. This means you need separate versions of the DCT in each CICS region in a CICSplex.

In CICS/ESA 4.1, the DFHDCT macro allows you to define a queue with both its local attributes and with the SYSIDNT of a remote CICS region that owns the queue.

CICS then handles the queue entry as either local or remote, depending on the SYSIDNT you have specified for it. If the SYSIDNT matches the SYSIDNT system initialization parameter of the CICS region in which the DCT is installed, the queue is treated as a local queue. If the SYSIDNT does not match the SYSIDNT system initialization parameter CICS treats the queue as a remote queue.

Benefits

This change is of benefit to systems programmers controlling a CICSplex.

In CICS/ESA 4.1, the system programmer's job is simplified by allowing a single table to be used in all the CICS systems in a CICSplex. Also, since the resource can be known both locally and remotely by the same name, applications are portable and can be moved around the CICSplex.

Changes to CICS externals

Table 47. DFHDCT TYPE=EXTRA

	DFHDCT	TYPE=EXTRA ,DESTID=name ,DSCNAME=name [,LENGTH=length] [,OPEN={**INITIAL**|DEFERRED{] [,RMTNAME=name] [,SYSIDNT=name]

LENGTH=length
Specifies, as a decimal value, the record length in bytes of fixed-length records in the queue. The length you specify must correspond to the RECSIZE length on the associated SDSCI entry in the DCT.

A CICS region that references a remote DCT entry requires the length of the record. If you do not specify it on the DCT entry, the application program must specify it on the WRITEQ and READQ requests.

If you omit the SYSIDNT parameter, LENGTH is ignored.

RMTNAME=name
Code this with the 1- to 4-character name by which the destination is known in the CICS region in which the destination resides (the remote region). If you omit this parameter, CICS uses the name specified on the DESTID parameter (that is, the local and remote names are the same).

This parameter is meaningful only when you specify the SYSIDNT parameter.

SYSIDNT=name
Identifies the CICS region in which the remote transient data queue resides. The 4-character alphanumeric name specified must match the SYSIDNT system initialization parameter specified on the region that "owns" the queue (the region in which the queue is a local resource).

If you omit SYSIDNT, the queue is treated as a local queue.

```
        DFHDCT   TYPE=INTRA
                 ,DESTID=name
                 [,DESTFAC={(TERMINAL[,trmidnt])
                           |FILE|(SYSTEM,sysidnt)}]
                 [,DESTRCV={NO|PH|LG}]
                 [,RMTNAME=name]
                 [,SYSIDNT=name]
                 [,TRANSID=name]
                 [,TRIGLEV={1|number}]
```

RMTNAME=name
The same as the RMTNAME parameter for extrapartition queues.

SYSIDNT=name
The same as the SYSIDNT parameter for extrapartition queues.

Selecting whether LLACOPY or BLDL is used to load programs

You can use the new system initialization parameter LLACOPY[8] to specify whether CICS is to use LLACOPY or the BLDL macro when locating modules in the DFHRPL concatenation. CICS uses the LLACOPY macro by default when searching for modules in the DFHRPL concatenation. However, the LLACOPY function issues an ENQ on SYSZLLA1.UPDATE, and CICS regions can be delayed while waiting for the ENQ if there are many tasks using the LLACOPY function on the same MVS image. The LLACOPY system initialization parameter makes CICS use of the LLACOPY macro optional. The syntax of the LLACOPY system initialization parameter is shown in Table 48, and described after that table. For details of all the system initialization parameters available in CICS/ESA 4.1, see the *CICS/ESA System Definition Guide*.

Table 48. LLACOPY system initialization parameter syntax

		[LLACOPY={**YES**|NO|NEWCOPY}]

LLACOPY={YES|NO|NEWCOPY}
Code this to indicate whether CICS is to use the LLACOPY macro or the BLDL macro when locating modules in the DFHRPL concatenation.

[8] The LLACOPY system initialization parameter was also added to CICS/ESA 3.3 by APAR PN40493.

YES
CICS always uses the LLACOPY macro when locating modules in the DFHRPL concatenation.

NO
CICS always uses the BLDL macro when locating modules in the DFHRPL concatenation.

NEWCOPY
CICS uses the LLACOPY only when a NEWCOPY or a PHASEIN is being performed. At all other times, CICS uses the BLDL macro when locating modules in the DFHRPL concatenation.

Notes:

1. If you code LLACOPY=NO or LLACOPY=NEWCOPY you can still benefit from having LLA managed data sets within your DFHRPL concatenation. Modules will continue to be loaded from VLF if appropriate.
2. If an LLA managed module has been altered, a BLDL macro may not return the new information and a subsequent load will still return the old copy of the module. To load the new module, an LLACOPY must be issued against that module or a MODIFY LLA,REFRESH command must be issued on a system console.

Increased CI size and number of buffers for temporary storage

In CICS/ESA 4.1, the following limits for CICS temporary storage have been increased:

- The maximum control interval (CI) size has been increased from 32KB to 64KB.

  The control interval size should be large enough to hold at least one (rounded up) temporary storage record, including 64 bytes of VSAM control information for control interval sizes less than, or equal to, 16 384, or 128 bytes of control information for larger control interval sizes.

- The maximum number of buffers has been increased from 255 to 32 767.

  You can implement this change by the TS system initialization parameter, as shown below.

Maximum number of temporary storage buffers (32K)

You can use the TS system initialization parameter to specify the number of CICS temporary storage buffers up to the maximum of 32 767. The number of buffers that you specify may have an effect on CICS performance, as described in "Performance considerations of TS and TD buffers" on page 510.

Table 49. The DFHSIT macro parameters

		[,TS=([COLD][,{0|**3**|value-1}][,{**3**|value-2}])]

TS=([COLD][,{0|3|decimal-value-1}][,{3|decimal-value-2}])
Specifies:

- Whether or not you want to cold start temporary storage

- The number of VSAM buffers to be used for auxiliary temporary storage
- The number of VSAM strings to be used for auxiliary temporary storage.

COLD
The type of start for the temporary storage program. If you do not want a cold start, code a comma before the second operand.

0 No buffers are required; that is, only MAIN temporary storage is required.

decimal-value-1
The number of buffers to be allocated for the use of auxiliary temporary storage. The value must be in the range 3 through 32 767.

CICS obtains, above 16MB, storage for the auxiliary temporary storage buffers in units of the page size (4KB). Because CICS optimizes the use of the storage obtained, TS may allocate more buffers than you specify, depending on the control interval (CI) size you have defined for the auxiliary temporary storage data set.

For example, if the CI size is 2048, and you specify 3 buffers (the default number), CICS actually allocates 4 buffers. This is because 2 pages (8192 bytes) are required to obtain sufficient storage for three 2048-byte buffers, a total of 6144 bytes, which would leave 2048 bytes of spare storage in the second page. In this case, CICS allocates another 2048-byte buffer to use all of the 8192 bytes obtained. In this way CICS makes use of storage that would otherwise be unavailable for any other purpose.

decimal-value-2
The number of VSAM strings to be allocated for the use of auxiliary temporary storage. The value must be in the range 1 through 255, and must not exceed the value specified in decimal-value-1. The default value is 3.

For more information about the TS system initialization parameter, see the *CICS/ESA System Definition Guide*.

Increased number of buffers for transient data

In CICS/ESA 4.1, the maximum number of buffers for transient data has been increased from 255 to 32 767.

You can use the TD system initialization parameter to specify the number of CICS transient data buffers up to the maximum of 32 767. The number of buffers that you specify may have an effect on CICS performance, as described in "Performance considerations of TS and TD buffers" on page 510.

Table 50. The DFHSIT macro parameters		
		[,TD=({**3**|number1}[,{**3**|number2}])]

TD=({3|decimal-value-1}[,{3|decimal-value-2}])
Specifies the number of VSAM buffers and strings to be used for intrapartition transient data (TD).

decimal-value-1
The number of buffers to be allocated for the use of intrapartition transient data. The value must be in the range 1 through 32 767. The default value is 3.

CICS obtains, above 16MB, storage for the TD buffers in units of the page size (4KB). Because CICS optimizes the use of the storage obtained, TD may allocate more buffers than you specify, depending on the control interval (CI) size you have defined for the intrapartition data set.

For example, if the CI size is 1536, and you specify 3 buffers (the default number), CICS actually allocates 5 buffers. This is because 2 pages (8192 bytes) are required to obtain sufficient storage for three 1536-byte buffers, a total of only 4608 bytes, which would leave 3584 bytes of spare storage in the second page. In this case, CICS allocates another 2 buffers (3072 bytes) to minimize the amount of unused storage. In this way CICS makes use of storage that would otherwise be unavailable for any other purpose.

decimal-value-2
The number of VSAM strings to be allocated for the use of intrapartition transient data. The value must be in the range 1 through 255, and must not exceed the value specified in decimal-value-1. The default value is 3.

For example, TD=(8,5) specifies 8 buffers and 5 strings.

The operands of the TD parameter are positional. You must code commas to indicate missing operands if others follow. For example, TD=(,2) specifies the number of strings and allows the number of buffers to default.

For more information about the TD system initialization parameter, see the *CICS/ESA System Definition Guide*.

Performance considerations of TS and TD buffers

When specifying the number of buffers for temporary storage and transient data, you should consider the following possible performance impacts:

- Using a large number of buffers means that for non-recoverable queues all processing can be performed without going to VSAM. This improves CICS performance. However, at shutdown all the buffers have to be flushed sequentially which can take a long time.
- If you specify a large number of buffers, and the number of queues is small, CICS takes longer to search down the chain of buffers for a particular queue.
- You can still specify only up to 255 VSAM strings. This means that there is no change on CICS waiting for VSAM strings.

Efficient deletion of shipped terminal definitions

This section describes the CICS/ESA 4.1 method of deleting redundant shipped terminal definitions.

Overview

In a transaction routing environment, terminal definitions can be "shipped" from a terminal-owning region (TOR) to an application-owning region (AOR) when they are first needed, rather than being statically defined in the AOR.

Note: The "terminal" could be an APPC device or system. In this case, the shipped definition would be of an APPC connection.

Shipped definitions can become redundant if:

- A terminal user logs off
- A terminal user stops using remote transactions
- The TOR is shut down
- The TOR is restarted, autoinstalled terminal definitions are not recovered, and the autoinstall user program, DFHZATDX, assigns a new set of termids to the same set of terminals.

At some stage redundant definitions must be deleted from the AOR (and from any intermediate systems between the TOR and AOR[9]). This is particularly necessary in the last case above, to prevent a possible mismatch between termids in the TOR and the back-end systems.

CICS/ESA 4.1 introduces a more efficient method for deleting redundant shipped definitions. It consists of two parts:

- Selective deletion
- A timeout delete mechanism.

Selective deletion

Each time a terminal definition is installed, CICS/ESA 4.1 creates a unique token and stores it within the terminal definition. Thus, if the definition is shipped to another region, the value of the token is shipped too. CICS passes the token on all transaction routing attach requests within the function management header (FMH). If, during attach processing, an existing shipped definition is found in the remote region, it is used *only if the token in the shipped definition matches that passed by the TOR*. Otherwise, it is deleted and an up-to-date definition shipped.

The timeout delete mechanism

You can use the timeout delete mechanism in your back-end systems, to delete shipped definitions that have not been used for transaction routing for a defined period.[10] Its purpose is to ensure that shipped definitions remain installed only while they are in use.

Timeout delete gives you flexible control over shipped definitions. CICS allows you to:

- Stipulate the minimum time a shipped definition must remain installed before being eligible for deletion
- Stipulate the time interval between invocations of the mechanism
- Reset these times online

---

[9] For brevity, we shall refer to AORs and intermediate systems collectively as "back-end systems".

[10] Shipped definitions are not deleted if there is an automatic initiate descriptor (AID) associated with the terminal.

- Cause the timeout delete mechanism to be invoked immediately.

Benefits

Some of the benefits of the new method are listed below.

Improved restart times

In CICS/ESA 3.3, if a TOR is restarted, a “remote reset” takes place, by which all shipped definitions in an attached back-end system are deleted. This happens even if terminal definitions are recovered in the TOR. All transaction routing requests are forced to wait until this mass deletion completes.

In CICS/ESA 4.1, selective deletion means that:

- If a TOR is restarted and terminal definitions are recovered, shipped definitions are *not* deleted and reshipped, but reused.
- If a TOR is restarted and terminal definitions are not recovered, shipped definitions are not deleted *en masse* but individually, as they are referenced by individual routed transactions. Thus, transaction routing requests do not have to wait until all shipped definitions have been deleted.

Fewer network flows

In CICS/ESA 3.3, when the user of an autoinstalled terminal logs off after having used transaction routing, “remote delete” requests are issued to delete the terminal definition from the back-end systems to which it had been shipped. However, there is no record of *which* back-end systems have received a specific definition. Therefore, if the TOR is attached to multiple systems, a remote delete request is sent to each one that has received any shipped definition. Remote delete requests can thus be transmitted, not only down the transaction routing path, but also to other interconnected systems, unnecessarily.

Similarly, after a restart of the TOR, a remote reset request can be transmitted to connected systems unnecessarily.

Furthermore, in CICS/ESA 3.3, if an intermediate system between a TOR and an AOR is restarted, that system too initiates a remote reset request to any back-end system that has received shipped definitions. With no change of terminal IDs in the TOR, these requests are redundant.

In CICS/ESA 4.1, the selective deletion and timeout delete mechanisms do away with these redundant flows.

More even performance

In CICS/ESA 3.3, the performance of some transactions (usually those routed to an AOR soon after a restart of the TOR) can be affected by a remote reset operation, while that of others (instigated after the operation has completed) is not affected at all. In CICS/ESA 4.1, the selective deletion mechanism ensures more even performance.

More accurate accounting

In CICS/ESA 3.3, the remote reset indicator is usually carried on the first transaction request to be routed to a back-end system after a restart of the TOR. This means that mass delete processing can be billed to the user transaction started in an AOR.

In CICS/ESA 4.1, the timeout delete mechanism runs as a CICS service transaction in back-end systems, and is billed accordingly.

Flexible control

Timeout delete gives you flexible control over the deletion of redundant definitions. For example, the parameters that control the mechanism allow you to arrange for a "tidy-up" operation to take place when the system is least busy. Your operators can use the CEMT transaction to modify the parameters online, or to invoke the mechanism immediately, should fine-tuning become necessary.

Implementing timeout delete

There are changes to CICS externals to enable you to use timeout delete in a CICS/ESA 4.1 region to which terminals are shipped. These are:

- Changes to system definition
- Changes to the system programming interface
- Changes to CICS-supplied transactions
- Changes to statistics.

The new system initialization parameters enable you to set time limits for the timeout delete mechanism.

Optionally, after CICS startup you can use a SET DELETSHIPPED command to reset these times, or a PERFORM DELETSHIPPED command to invoke the timeout delete mmediately.

Changes affecting system definition

Table 51 shows the system initialisation parameters that you specify to enable the timeout delete mechanism.

Table 51. DFHSIT parameters for timeout delete

	DFHSIT	TYPE={**CSECT**|DSECT} ... [,DSHIPIDL={**020000**|hhmmss}] [,DSHIPINT={**120000**|hhmmss}] ...
	END	DFHSITBA

DSHIPIDL={020000|hhmmss}
Specifies the minimum time, in hours, minutes, and seconds, that an *inactive* shipped terminal definition must remain installed in this region. When the timeout delete mechanism is invoked, only those shipped definitions that have been inactive for longer than the specified time are deleted.

You can use this parameter in a transaction routing environment, on the application-owning and intermediate regions, to prevent terminal definitions having to be reshipped because they have been deleted prematurely.

The default minimum idle time is 2 hours.

hhmmss Specify a 1 to 6 digit number in the range 0–995959. Numbers that have fewer than six digits are padded with leading zeros.

DSHIPINT={120000|0|hhmmss}

Specifies the interval between invocations of the timeout delete mechanism. The timeout delete mechanism removes any shipped terminal definitions that have not been used for longer than the time specified by the DSHIPIDL parameter.

You can use this parameter in a transaction routing environment, on the application-owning and intermediate regions, to control:

- How often the timeout delete mechanism is invoked.
- The approximate time of day at which a mass delete operation is to take place, relative to CICS startup.

  Note: For more flexible control over when mass delete operations take place, you can use a CEMT SET DELETSHIPPED or EXEC CICS SET DELETSHIPPED command to reset the interval. (The revised interval starts *from the time the command is issued*, **not** from the time the remote delete mechanism was last invoked, nor from CICS startup.)

0 The timeout delete mechanism is not invoked. You might set this value in a terminal-owning region, or if you are not using shipped definitions.

hhmmss Specify a 1 to 6 digit number in the range 1–995959. Numbers that have fewer than six digits are padded with leading zeros.

Changes to the system programming interface

General-use programming interface

There are three new system programming commands—EXEC CICS INQUIRE, PERFORM and SET DELETSHIPPED.

INQUIRE DELETSHIPPED

Function

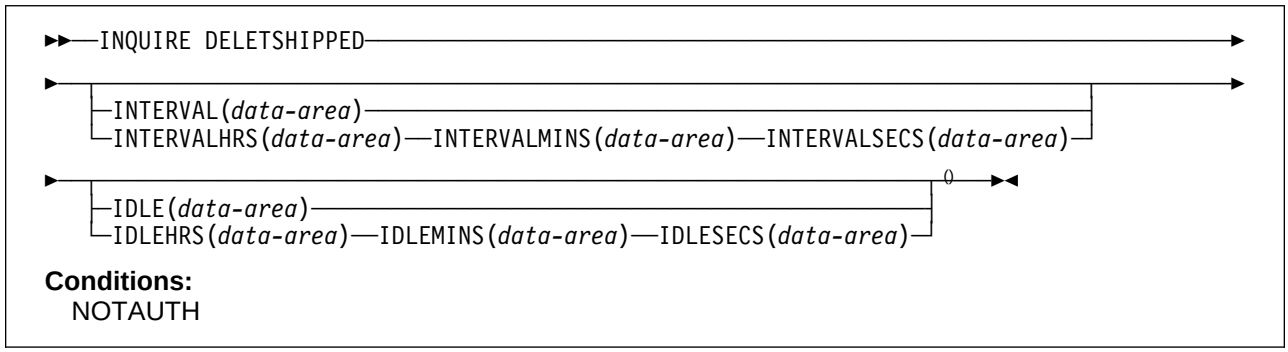
The INQUIRE DELETSHIPPED command returns system settings that control the CICS timeout delete mechanism.

Description

The CICS timeout delete mechanism is invoked at user-specified intervals to remove any shipped terminal definitions that have not been used for longer than a user-specified time.

INQUIRE DELETSHIPPED displays the current settings of the parameters that control the mechanism.

Syntax



INQUIRE DELETSHIPPED options

IDLE(data-area)
returns, as a 4-byte packed decimal value in the form "**0hhmmss**+", the minimum time that an *inactive* shipped terminal definition must remain installed in this region. When the CICS timeout delete mechanism is invoked, only those shipped definitions that have been inactive for longer than this time are deleted.

The time interval can be in the range 00–99 hours; 00–59 minutes; and 00–59 seconds.

Note: If you do not use this option but prefer to use the individual IDLEHRS, IDLEMINS, and IDLESECS options instead, you must use all three individual options.

IDLEHRS(data-area)
returns, as a fullword binary value, the number of hours for which an inactive shipped terminal definition must remain installed in this region. Values are in the range 0–99.

IDLEMINS(data-area)
returns, as a fullword binary value, the number of minutes beyond the hour value returned in IDLEHRS for which an inactive shipped terminal definition must remain installed in this region. Values are in the range 0–59.

IDLESECS(data-area)
returns, as a fullword binary value, the number of seconds beyond the minute value returned in IDLEMINS for which an inactive shipped terminal definition must remain installed in this region. Values are in the range 0–59.

INTERVAL(data-area)
returns, as a 4-byte packed decimal value in the form "**0hhmmss**+", the interval between invocations of the CICS timeout delete mechanism. The timeout delete mechanism removes any shipped terminal definitions that have not been used for longer than the time returned by the IDLE option.

The time interval can be in the range 00–99 hours; 00–59 minutes; and 00–59 seconds.

Note: If you do not use this option but prefer to use the individual INTERVALHRS, INTERVALMINS, and INTERVALSECS options instead, you must use all three individual options.

INTERVALHRS(data-area)
returns, as a fullword binary value, the number of hours between invocations of the timeout delete mechanism. Values are in the range 0–99.

INTERVALMINS(data-area)
returns, as a fullword binary value, the number of minutes beyond the hour value returned in INTERVALHRS between invocations of the timeout delete mechanism. Values are in the range 0–59.

INTERVALSECS(data-area)
returns, as a fullword binary value, the number of seconds beyond the minite value returned in INTERVALMINS between invocations of the timeout delete mechanism. Values are in the range 0–59.

Conditions

Condition	*RESP2*	*Meaning*
NOTAUTH	100	The use of this command is not authorized.

PERFORM DELETSHIPPED

Function

This causes the CICS timeout delete mechanism to be invoked immediately.

Description

The CICS timeout delete mechanism removes any shipped terminal definitions that have not been used for longer than the time specified on the DSHIPIDL system initialization parameter, or on a subsequent SET DELETSHIPPED IDLE command. The interval between scheduled invocations of the mechanism is specified on the DSHIPINT system initialization parameter, or on a subsequent SET DELETSHIPPED INTERVAL command.

PERFORM DELETSHIPPED invokes timeout delete immediately.

Notes:

1. Shipped definitions are not deleted if there is an automatic initiate descriptor (AID) associated with the terminal.
2. Issuing a PERFORM DELETSHIPPED command does **not** reset the start of the time interval until the next invocation of the timeout delete mechanism. In other words, causing the timeout delete mechanism to be invoked immediately does not affect the time remaining until the next scheduled invocation.

Syntax

Conditions:
NOTAUTH

PERFORM DELETSHIPPED options

None.

Conditions

Condition	*RESP2*	*Meaning*
NOTAUTH	100	The use of this command is not authorized.

SET DELETSHIPPED

Function

This allows you to change system settings that control the timeout delete mechanism used to remove redundant shipped terminal definitions.

Description

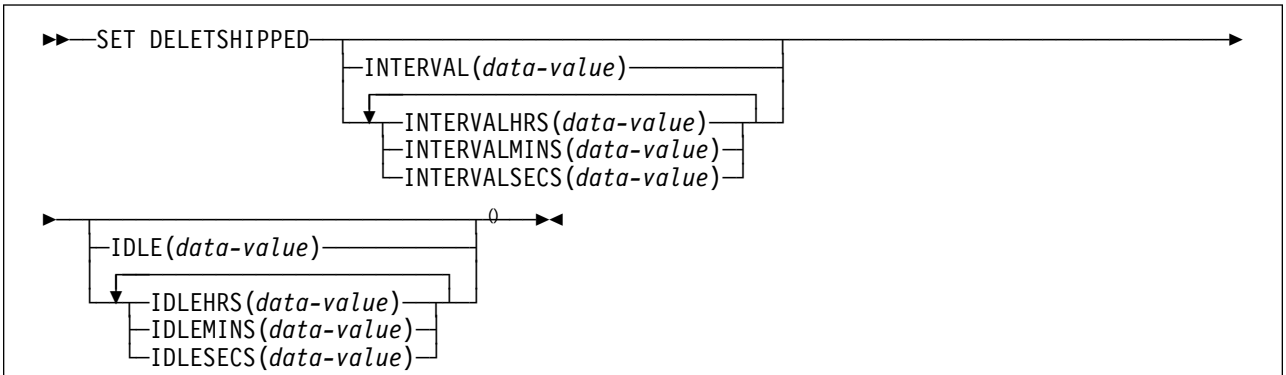
SET DELETSHIPPED allows you to stipulate:

- The minimum time a shipped terminal definition must remain installed before being eligible for deletion by the timeout delete mechanism
- The time interval between invocations of the timeout delete mechanism.

The command allows you to specify time intervals in three ways. For example, to specify the minimum time a shipped definition must remain installed, you could use:

1. IDLE(hhmmss), with values in the ranges IDLE(0–99 0–59 0–59). IDLE(010203) means one hour, two minutes, and three seconds.
2. A combination of at least two of IDLEHRS(0–99), IDLEMINS(0–59), and IDLESECS(0–59). IDLEHRS(1) IDLESECS(3) means one hour and three seconds (the minutes default to zero).
3. One of IDLEHRS(0–99), IDLEMINS(0–5999), or IDLESECS(0–359 999). IDLEHRS(1) means one hour. IDLEMINS(62) means one hour and two minutes. IDLESECS(3723) means one hour, two minutes, and three seconds.

Syntax



Conditions:
INVREQ, NOTAUTH

SET DELETSHIPPED options

IDLE(data-value)
specifies, as a 4-byte packed decimal value in the form "**0hhmmss**+", the minimum time that an *inactive* shipped terminal definition must remain installed in this region. When the CICS timeout delete mechanism is invoked, only those shipped definitions that have been inactive for longer than this time are deleted.

The time interval can be in the range 00–99 hours; 00–59 minutes; and 00–59 seconds.

At CICS startup, the interval is set to the value specified on the DSHIPIDL system initialization parameter.

IDLEHRS(data-value)
specifies, as a fullword binary value, the number of hours for which inactive shipped terminal definition must remain installed in this region. Values must be in the range 0–99.

IDLEMINS(data-value)
specifies, as a fullword binary value, the number of minutes for which inactive shipped terminal definition must remain installed in this region.

If used with IDLEHRS and IDLESECS, values must be in the range 0–59. If used on its own, values must be in the range 0–5999.

IDLESECS(data-value)
specifies, as a fullword binary value, the number of seconds for which inactive shipped terminal definition must remain installed in this region.

If used with IDLEHRS and IDLEMINS, values must be in the range 0–59. If used on its own, values must be in the range 0–359999.

INTERVAL(data-value)
specifies, as a 4-byte packed decimal value in the form "**0hhmmss**+", the interval between invocations of the CICS timeout delete mechanism. The timeout delete mechanism removes any shipped terminal definitions that have not been used for longer than the time specified by the IDLE option.

The time interval can be in the range 00–99 hours; 00–59 minutes; and 00–59 seconds. If you specify 0, the timeout delete mechanism is not invoked.

At CICS startup, the interval is set to the value specified on the DSHIPINT system initialization parameter. By resetting the interval, you can change the time of day at which a mass delete operation takes place.

Note: The revised interval starts from *the time the command is issued*, **not** from the time the timeout delete mechanism was last invoked, nor from the time of CICS startup.

INTERVALHRS(data-value)
specifies, as a fullword binary value, the number of hours between invocations of the timeout delete mechanism. Values must be in the range 0–99.

INTERVALMINS(data-value)
specifies, as a fullword binary value, the number of minutes between invocations of the timeout delete mechanism.

If used with INTERVALHRS and INTERVALSECS, values must be in the range 0–59. If used on its own, values must be in the range 0–5999.

INTERVALSECS(data-value)
specifies, as a fullword binary value, the number of seconds between invocations of the timeout delete mechanism.

If used with INTERVALHRS and INTERVALMINS, values must be in the range 0–59. If used on its own, values must be in the range 0–359999.

Conditions

Condition	*RESP2*	*Meaning*
INVREQ	1	INTERVAL value is invalid.
INVREQ	2	INTERVALHRS value is not in the range 0–99.
INVREQ	3	INTERVALMINS value is invalid.
INVREQ	4	INTERVALSECS value is invalid.
INVREQ	5	IDLE value is invalid.
INVREQ	6	IDLEHRS value is not in the range 0–99.
INVREQ	7	IDLEMINS value is invalid.
INVREQ	8	IDLESECS value is invalid.
NOTAUTH	100	The use of this command is not authorized.

End of General-use programming interface

Changed CICS supplied transactions

There are three new CEMT commands—INQUIRE DELETSHIPPED, PERFORM DELETSHIPPED, and SET DELETSHIPPED.

CEMT INQUIRE DELETSHIPPED

This new command displays the system parameters that control the CICS timeout delete mechanism.

Description

The CICS timeout delete mechanism is invoked at user-specified intervals to remove any shipped terminal definitions that have not been used for longer than a user-specified time.

INQUIRE DELETSHIPPED displays the current settings of the parameters that control the mechanism.

Input:

Press the Clear key and type `CEMT INQUIRE DELETSHIPPED` (the minimum abbreviation is `CEMT I DE`). You will get a display screen.

To change attributes, you can:

- Overtype your changes on the INQUIRE screen after tabbing to the appropriate field.
- Use the CEMT SET DELETSHIPPED command.

Displayed fields

Idle(value)

displays, in the form "**0hhmmss**+", the minimum time that an *inactive* shipped terminal definition must remain installed in this region. When the CICS timeout delete mechanism is invoked, only those shipped definitions that have been inactive for longer than this time are deleted.

The time interval can be in the range 00–99 hours; 00–59 minutes; and 00–59 seconds.

Interval**(value)**

displays, in the form "**0hhmmss**+", the interval between invocations of the CICS timeout delete mechanism. The timeout delete mechanism removes any shipped terminal definitions that have not been used for longer than the time displayed by the IDLE option.

The time interval can be in the range 00–99 hours; 00–59 minutes; and 00–59 seconds.

CEMT PERFORM DELETSHIPPED

This new command causes CICS to invoke the timeout delete mechanism immediately.

Description:

The CICS timeout delete mechanism removes any shipped terminal definitions that have not been used for longer than the time specified on the DSHIPIDL system initialization parameter, or on a subsequent SET DELETSHIPPED IDLE command. The interval between scheduled invocations of the mechanism is specified on the DSHIPINT system initialization parameter, or on a subsequent SET DELETSHIPPED INTERVAL command.

PERFORM DELETSHIPPED invokes timeout delete immediately.

Notes:

1. Shipped definitions are not deleted if there is an automatic initiate descriptor (AID) associated with the terminal.
2. Issuing a PERFORM DELETSHIPPED command does **not** reset the start of the time interval until the next invocation of the timeout delete mechanism. In other words, causing the timeout delete mechanism to be invoked immediately does not affect the time remaining until the next scheduled invocation.

Syntax:

Options

None.

CEMT SET DELETSHIPPED

This new command enables you to change the system parameters that control the CICS timeout delete mechanism.

Description

The CICS timeout delete mechanism is invoked at user-specified intervals to remove any shipped terminal definitions that have not been used for longer than a user-specified time.

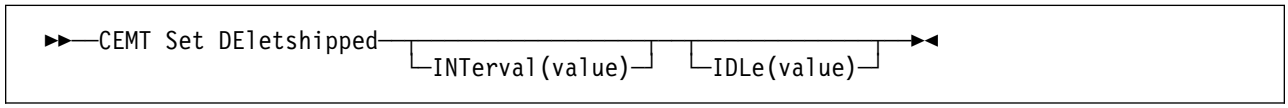
SET DELETSHIPPED allows you to change the values that control the mechanism.

Input:

Press the Clear key to clear the screen. There are two ways of commencing this transaction:

- Type `CEMT SET DELETSHIPPED` (the minimum abbreviation is `CEMT S DE`). You get a display that lists the current status, similar to that obtained by CEMT INQUIRE DELETSHIPPED. You can then tab to the highlighted or blank fields and overtype them with the required values.
- Type `CEMT SET DELETSHIPPED` (`CEMT S DE`), followed by one or more attributes that you wish to change. For example, `cemt s de idl(0150000)` specifies that shipped terminal definitions are to remain installed for at least fifteen hours after they become inactive.

Syntax:



Options

Idle(value)

specifies, in the form "**0hhmmss**+", the minimum time that an *inactive* shipped terminal definition must remain installed in this region. When the CICS timeout delete mechanism is invoked, only those shipped definitions that have been inactive for longer than this time are deleted.

The time interval can be in the range 00–99 hours; 00–59 minutes; and 00–59 seconds.

At CICS startup, the interval is set to the value specified on the DSHIPIDL system initialization parameter.

Interval**(value)**

specifies, in the form "**0hhmmss**+", the interval between invocations of the CICS timeout delete mechanism. The timeout delete mechanism removes any shipped terminal definitions that have not been used for longer than the time specified by the IDLE option.

The time interval can be in the range 00–99 hours; 00–59 minutes; and 00–59 seconds. If you specify 0, the timeout delete mechanism is not invoked.

At CICS startup, the interval is set to the value specified on the DSHIPINT system initialization parameter. By resetting the interval, you can change the time of day at which a mass delete operation takes place.

Note: The revised interval starts from *the time the command is issued*, **not** from the time the timeout delete mechanism was last invoked, nor from the time of CICS startup.

Changes to statistics

"Shipped remote definitions: Global statistics" shows the statistics produced for shipped terminal definitions.

Shipped remote definitions: Global statistics

These statistics are available online, and are mapped by the DFHA04DS DSECT.

DFHSTUP name	Field name	Description
Delete shipped interval	A04RDINT	is the currently-specified time delay, in the form **hhmmss**, between invocations of the timeout delete mechanism that removes redundant shipped terminal definitions. The value is set either by the DSHIPINT system initialization parameter, or by a subsequent SET DELETSHIPPED command.
		Reset characteristic: not reset.
Delete shipped idle time	A04RDIDL	is the currently-specified minimum time, in the form **hhmmss**, that an inactive shipped terminal definition must remain installed in this region, before it becomes eligible for removal by the CICS timeout delete mechanism. The value is set either by the DSHIPIDL system initialization parameter, or by a subsequent SET DELETSHIPPED command.
		Reset characteristic: not reset.
Shipped terminals built	A04SKBLT	is the number of shipped remote terminal definitions built during the recording period.
		Reset characteristic: reset to zero.
Shipped terminals installed	A04SKINS	is the number of shipped remote terminal definitions currently installed in this region.
		Reset characteristic: not reset.
Shipped terminals timed out	A04SKDEL	is the number of shipped remote terminal definitions deleted by the timeout delete mechanism during the recording period.
		Reset characteristic: reset to zero.
Times interval expired	A04TIEXP	is the number of times the shipped delete interval between invocations of the timeout delete mechanism expired during the recording period.
		Reset characteristic: reset to zero.
Remote deletes received	A04RDREC	is the number of old-style (pre-CICS/ESA 4.1) remote delete instructions received by this region since the start of the recording period.
		Reset characteristic: reset to zero.
Remote deletes issued	A04RDISS	is the number of old-style (pre-CICS/ESA 4.1) remote delete instructions issued by this region since the start of the recording period.
		Reset characteristic: reset to zero.

DFHSTUP name	Field name	Description
Successful remote deletes	A04RDDEL	is the number of shipped terminal definitions deleted from this region because of old-style remote delete instructions, since the start of the recording period.
		Reset characteristic: reset to zero.
Current idle count	A04CIDCT	is the number of shipped terminal definitions that are not currently being used by a running transaction.
		Reset characteristic: not reset.
Current idle time	A04CIDLE	is the total time, for all currently-idle shipped terminal definitions, since each was last used.
		Reset characteristic: not reset.
Current maximum idle time	A04CMAXI	is the maximum time since a currently-idle shipped terminal definition was last used.
		Reset characteristic: not reset.
Total idle count	A04TIDCT	is the total number of times, for all shipped terminal definitions, that each has been idle during the recording period.
		Note: This figure excludes the number of those currently idle (Current idle count).
		Reset characteristic: reset to zero.
Total idle time	A04TIDLE	is the total time, for all shipped terminal definitions, for which each has been idle during the recording period.
		Note: This figure excludes the time since currently-idle definitions were last used (Current idle time).
		Reset characteristic: reset to zero.
Maximum idle time	A04TMAXI	is the maximum time for which a shipped terminal definition was idle during the recording period.
		Note: This figure does not take into account the times since currently-idle definitions were last used.
		Reset characteristic: reset to zero.

Changes to global user exits

There are no changes to global user exits. However, you should be aware of new circumstances in which the existing exits, XALTENF and XICTENF, could be invoked. XALTENF and XICTENF are called when the “terminal not known” condition occurs. This happens if an automatic transaction initiation (ATI) request attempts to start a transaction that is associated with a remote terminal that is not defined to this system. Usually, the terminal is not defined because you are using shippable definitions and no transaction routing has yet taken place for the terminal.

Be aware that if you allow temporarily inactive shipped definitions too short a life (by setting too low a value for the DSHIPIDL system initialization parameter or the IDLE option of the SET DELETSHIPPED command) you could increase the number of calls to XALTENF and XICTENF.

New abend codes

The following new abend codes are introduced.

AZVM

Explanation: An unexpected error has occurred in DFHZATMF. This is probably caused by DFHZATMF being unable to address the CSA, EIB or the TCA.

System Action: The task is abnormally terminated with a CICS transaction dump.

User Response: Ensure that DFHZATMF is not being called incorrectly. If this is not the cause of the abend, this is a CICS logic error.

Module: DFHZATMF

AZVN

Explanation: The remote delete flag transaction of DFHZATMF (CRMF) has been started directly from a terminal. This is not permitted. This transaction can only be started internally by CICS.

System Action: The transaction is abnormally terminated with a transaction dump.

User Response: None.

Module: DFHZATMF

AZVO

Explanation: The remote delete transaction of DFHZATMD (CRMD) has been started directly from a terminal. This is not permitted. This transaction can only be started internally by CICS.

System Action: The transaction is abnormally terminated with a transaction dump.

User Response: None.

Module: DFHZATMD

AZVP

Explanation: An unexpected error has occurred in DFHZATMD. This is probably caused by DFHZATMD being unable to address the CSA, EIB or the TCA.

System Action: The task is abnormally terminated with a CICS transaction dump.

User Response: Ensure that DFHZATMD is not being called incorrectly. If this is not the cause of the abend, this is a CICS logic error.

Module: DFHZATMD

Performance

The selective deletion and timeout delete mechanisms result in:

- A considerable reduction in the pathlength of some transactions. See “More even performance” on page 512.
- A reduction in the number of network flows, leading to better system performance, particularly across a complex of CICS/ESA 4.1 systems. See “Fewer network flows” on page 512.
- Depending on your choice of DSHIPINT and DSHIPIDL settings, a possible reduction in the number of mass deletions of shipped definitions, and a scheduling of those that do take place for times when your system is lightly loaded.

Note that a poor choice of values for DSHIPINT and DSHIPIDL could result in unnecessary mass delete operations. Here are some suggestions for coding these parameters:

DSHIPIDL: In setting this value, you must consider the length of the work periods during which remote users access resources on this system. Do they access the system intermittently, all day? Or is their work concentrated into intensive, shorter periods?

By setting too low a value, you could cause definitions to be deleted and reshipped unnecessarily. It is also possible that you could cause ATI requests to fail (see "Changes to global user exits" on page 523).

DSHIPINT: You can use this value to control the time of day at which your mass delete operations take place. For example, if you usually warm-start CICS at 7 a.m., you could set DSHIPINT to 150000, so that the timeout delete mechanism is invoked at 10 p.m., when few users are accessing the system.

Warning: If CICS is recycled, perhaps because of a failure, the timeout delete interval is reset. Continuing the previous example, if CICS is recycled at 8:00p.m., the timeout delete mechanism will be invoked at 11:00a.m. the following day (15 hours from the time of CICS initialization). In these circumstances, you could use the SET DELETSHIPPED and PERFORM DELETSHIPPED commands to accurately control when a timeout delete takes place.

Migration considerations

For compatibility with previous releases of CICS, CICS/ESA 4.1 continues to support the old remote delete and remote reset mechanisms. You can always use the new timeout delete mechanism on any CICS/ESA 4.1 back-end sysytem. Whether the new selective deletion mechanism or the old-style remote delete and reset operates depends on the level of the front-end system. For example, consider the following combinations of front- and back-end systems.

Note: A "front-end" can be a TOR or an intermediate system. Likewise, a "back-end" can be an AOR or an intermediate system.

CICS/ESA 4.1 (or later) front-end to CICS/ESA 4.1 (or later) back-end

You can use timeout delete on the back-end system.

Based on the unique tokens passed by the front-end system, the back-end uses selective deletion to remove redundant definitions singly, as they are referenced by routed transactions.

CICS/ESA 4.1 (or later) front-end to pre-CICS/ESA 4.1 back-end

You cannot use timeout delete on the back-end system.

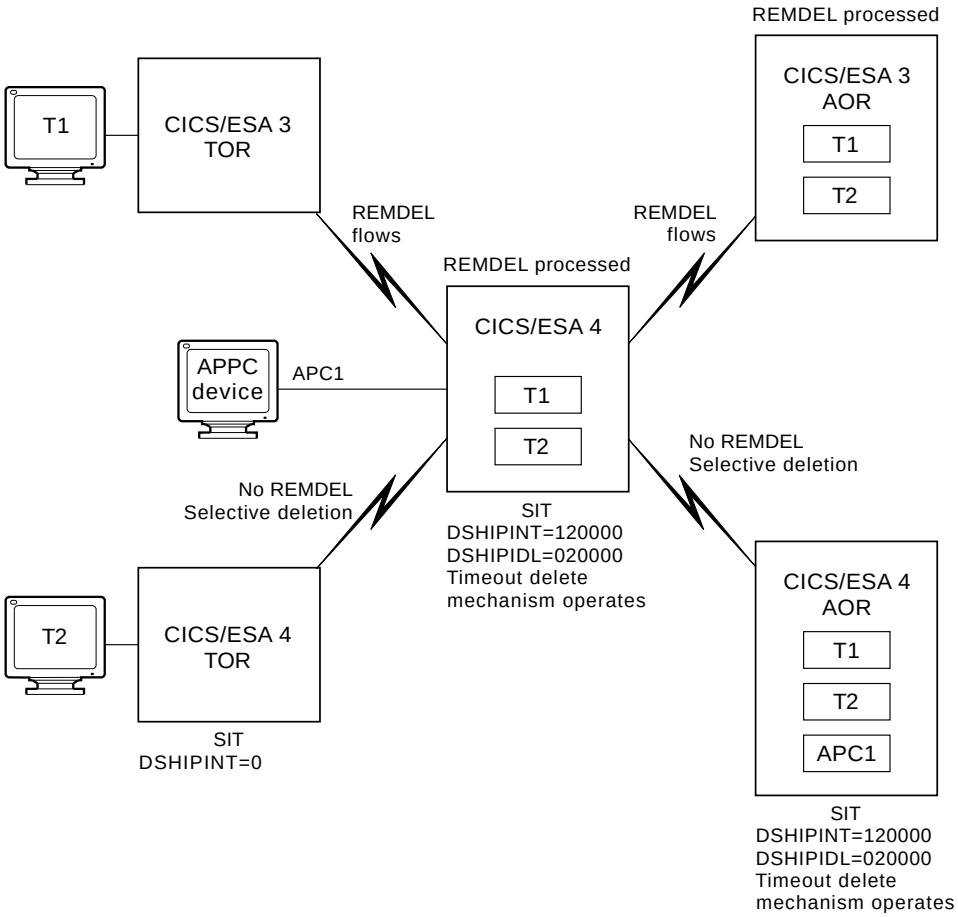
The front-end system uses the old-style remote delete and remote reset mechanisms. Thus *all* shipped definitions in the back-end system—whether redundant or not—are deleted after a restart of the TOR or of an intermediate system.

Pre-CICS/ESA 4.1 front-end to CICS/ESA 4.1 (or later) back-end

You can use timeout delete on the back-end system. The front-end system uses the old-style remote delete and remote reset mechanisms, which are honored by the back-end system.

Note: If you migrate a pre-CICS/ESA 4.1 system to CICS/ESA 4.1, any other CICS/ESA 4.1 (or later) systems to which it is connected will not recognize the upgrade (and therefore continue to issue old-style remote delete and remote reset requests) until their connections to the upgraded system are reinstalled.

Figure 40 shows various combinations of front- and back-end systems. In the figure, old-style remote delete and remote reset requests are shown collectively as "REMDEL"s.



KEY:

REMDEL *Pre-4.1 remote reset and remote delete requests*

T1
T2 *Remote terminal definitions*

APC1 *Remote APPC connection definition*

Figure 40. Deletion of shipped terminal definitions in a mixed-release network

Extensions to the DFHTST macros

APAR PQ00933

The TST changes described here are supplied by the PTF for APAR PQ00933, which is planned for shipment in April 1997.

The temporary storage table (TST) macros are extended in the following ways:

- The TYPE=LOCAL macro is added to enable you to specify generic names for local queues.
- A special null operand, of the form (), is allowed on the DATAID parameters of all the DFHTST TYPE= macros. This defines a special generic name that matches all queues not otherwise defined by more specific names on other DATAID parameters.
- The DATAID parameter on the TYPE=REMOTE macro is changed to allow the list form of generic queue names. The syntax for DATAID on *all* TST entries becomes:

  DATAID=(*character-string[,character-string,...])|()*
- The special queues that CICS creates for the REQID parameter on START commands no longer cause any conflict with generic names defined in TYPE=REMOTE macro entries. When processing TS queues created for REQIDs, CICS treats them automatically as local, and does not search the TST.

For details of the TYPE=LOCAL macro, and other associated changes, see the *CICS/ESA Resource Definition Guide*.

Chapter 32. Changes for installation, operations, and customization

This chapter describes a number of changes in CICS/ESA 4.1 that affect installation, operations, and customization. These items are:

- Application programming language features
- Changes to the CICS local catalog data set
- Changes to the XEIIN and XEIOUT global user exits
- Changes to the XTCATT global user exit
- New CEMT command to inquire on temporary storage queues
- Message DFHFC0988 for recall of migrated VSAM data sets
- Sample shut-down assist program for use in CICS shutdown.

Application programming languages features

This section outlines the changes in CICS/ESA 4.1 to distribute and install CICS application programming languages as separate SMP/E features.

In CICS/ESA 4.1, support for the following application programming languages:

COBOL (OS/VS COBOL and VS COBOL II)
PL/1
C/370

is provided as separate SMP/E features of CICS, but still installed as part of the installation process for CICS/ESA 4.1. These features are summarized in Table 52.

Table 52. CICS/ESA 4.1 application programming language features

Language	Feature (FMID)	Distribution library
+ COBOL	JCI4101	ADFHCOB and ADFHSAMP
+ PL/1	JCI4102	ADFHPL1 and ADFHSAMP
+ C/370	JCI4103	ADFHC370 and ADFHSAMP

Changes to the CICS local catalog data set

You must define and initialize new catalog data sets for each CICS/ESA 4.1 region that you are initializing for the first time. There are sample job streams shown in the *CICS/ESA System Definition Guide* that you can use for defining the catalogs.

APAR PQ07674

Information added for APAR PQ07674 November 1998.

Use the CICS-supplied utility program, DFHSMUTL, to add records to enable the CICS self-tuning mechanism for storage manager domain subpools.

There is also DFHDEFDS, a CICS-supplied skeleton job in the CICS410.SDFHINST library that you can tailor to create CICS system data sets. You should avoid using old JCL because there may be changes to some data set parameters. For example, the maximum record size of the local catalog is increased in CICS/ESA 4.1 to 124 bytes. The recommended RECORDSIZE parameter values for the definition of the local catalog data set are as follows:

```
RECORDSIZE( 50 150 )
```

This record size allows for any increase that may occur, perhaps as a result of any service that may be applied.

The number of records written to the local catalog data set is also increased in CICS/ESA 4.1. The sample job shown in Figure 41 shows recommended values for the size-related parameters.

```
//LOCAT    JOB accounting info,,CLASS=A
//DEFLCD   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
//*
         DEFINE CLUSTER -
                (NAME( CICS410.applid.DFHLCD)  -
                 INDEXED                       -
                 RECORDS( 300 10 )             -
                 FREESPACE(10 10)              -
                 SHAREOPTIONS( 2 )             -
                 REUSE                         -
                 VOLUMES( volid ))             -
           DATA                                -
                (NAME( CICS410.applid.DATA.DFHLCD )   -
                 KEYS( 28 0 )                  -
                 RECORDSIZE( 50 150 )          -
                 CONTROLINTERVALSIZE( 2048 ))  -
           INDEX (NAME( CICS410.applid.INDEX.DFHLCD ) -
                  IMBED                        -
                  REPLICATE )
/*
```

Figure 41. Sample job to define and initialize the local catalog

Changes to the EXEC interface program exits XEIIN and XEIOUT

Product-sensitive programming interface

XEIIN and XEIOUT are global user exit points in the EXEC interface program. XEIIN is invoked before the execution of an EXEC CICS command. The sequence of events is changed in CICS/ESA 4.1, and is as follows:

TRACE – XEIIN – EDF – command

XEIOUT is invoked after execution of an EXEC CICS command. In this case, the sequence of events is the same as in CICS/ESA 3.3, as follows:

command – EDF – XEIOUT – TRACE

Bypassing commands

You can set UERCBYP from within XEIIN to bypass execution of a command. If you use bypass, EDF is not invoked, but XEIOUT and exit trace are invoked if they are active.

Other changes

The other changes to the EXEC interface exits in CICS/ESA 4.1 are as follows:

- There are new parameters passed in the exit-specific parameter list, and the parameter list is now the same for both XEIIN and XEIOUT.
- On entry to both XEIIN and XEIOUT, the UEPARG parameter contains the address of the command parameter list.
- The UEPGROUP parameter is obsolete in XEIOUT, which means that any global user exit programs written for the XEIOUT exit, and which use the old UEPGROUP parameter, must be modified.

As in earlier releases, the command parameter list, addressed by UEPARG, contains a string of command data known as **argument 0**. For information about how to use argument 0 to interpret the command parameter list, see the *CICS/ESA Customization Guide*.

WARNING

Modifying CICS commands by tampering with argument 0 is ***not*** supported, and leads to unexpected errors or results.

For example, if an application program is written in assembler or PL/I and you modify argument 0, you will be writing to program storage (that is, storage occupied by the program itself), which could cause 0C4 abends. Furthermore, modifying argument 0 not only alters the CICS command for *this execution* of the command in the application program, it changes the CICS command in the virtual storage copy of the application program. This means that the next task to invoke the same copy of the program will also execute the modified command.

This particular example of the danger of tampering with argument 0 does not apply to COBOL or C/370 application programs, but nevertheless you should not modify CICS commands for application programs written in any supported language.

Table 53 on page 532 and Table 54 on page 533 show the parameters received by XEIIN and XEIOUT.

Exit XEIIN

Table 53. XEIIN exit-specific parameters

	Invoked before the execution of an EXEC CICS command.	
Exit-specific parameters	**UEPARG**	Address of the EXEC command parameter list.
	UEPEXECB	Address of the system EIB.
	UEPUSID	Address of the 8-character userid.
	UEPPGM	Address of the 8-character application program name.
	UEPLOAD	Actual address of the application program load-point.
	UEPRSA	Actual address of the application register save area.
Return codes	**UERCNORM**	Continue processing.
	UERCBYP	Bypass the execution of this command.
	UERCPURG	Task purged during XPI call.
XPI calls	All XPI calls are supported in this exit.	

UEPRSA
The actual address of a register save area of the application program that issued the EXEC CICS command. This save area contains the contents of the registers at the point when the program issued the EXEC CICS command.

UEPLOAD
The actual load point address of the application program.

The other parameters are pointers to storage areas that contain the parameter values described in Table 53 and Table 54 on page 533.

Exit XEIOUT

Table 54. XEIOUT exit-specific parameters

	Invoked after the execution of an EXEC CICS command.	
Exit-specific parameters	**UEPARG**	Address of the EXEC command parameter list.
	UEPEXECB	Address of the system EIB.
	UEPUSID	Address of the 8-character userid.
	UEPPGM	Address of the 8-character application program name.
	UEPLOAD	Actual address of the application program load-point.
	UEPRSA	Actual address of the application register save area.
Return codes	**UERCNORM**	Continue processing.
	UERCPURG	Task purged during XPI call.
	No other return codes are supplied.	
XPI calls	All XPI calls are supported in this exit.	

Changes to the XTCATT global user exit

There is a new exit-specific parameter on the terminal control program global user exit, XTCATT. The new parameter is UEPTRAN, which contains the address of the 4-byte transaction identifier.

Note: The global user exit program can modify the transaction id in the XTCATT (and in the XZCATT user exits).

Exit XTCATT

	Invoked before task attach.	
Exit-specific parameters	**UEPTCTTE**	Address of the terminal control table terminal entry (TCTTE). The TCTTE can be mapped using the DSECT DFHTCTTE.
	UEPTIOA	Address of the terminal input/output area (TIOA). The TIOA can be mapped using the DSECT DFHTIOA. However, fields TIOASAL and TIOASCA are **not** programming interfaces.
	UEPTCTLE	Address of the terminal control table line entry (TCTLE). The TCTLE can be mapped using the DSECT DFHTCTLE.
	UEPTRAN	Address of the 4-byte transaction id.
Return codes	**UERCNORM**	Continue processing.
XPI calls	All can be used.	

New loader domain global user exits XLDLOAD and XLDELETE

APAR PQ19300

informaiton added for APAR PQ19300 November 1998

There are two global user exits added to the loader domain, XLDLOAD and XLDELETE. CICS invokes XLDLOAD when a new copy of a program has been loaded into storage, and before it is made available for use. XLDELETE is invoked after a copy of a program is released by CICS and just before the program storage is freed. In the case of LPA-resident programs, the exits are also invoked when a program is acquired or released, although there is no physical load, or storage to free.

These loader domain exit points are for providing information only, and no changes to the global user exit parameter lists are permitted. Any changes made to parameters are ignored by CICS, as is any return code set by an exit program. You cannot issue any API or SPI commands, or make any XPI calls at these exit points.

For details of the information provided by these exits, see the *CICS/ESA Customization Guide*.

End of Product-sensitive programming interface

New CEMT command to inquire on temporary storage queues

The CEMT INQUIRE TSQUEUE is introduced in CICS/ESA 4.1 to provide equivalent function to the SPI command (EXEC CICS INQUIRE TSQUEUE) which was new in CICS/ESA 3.3.

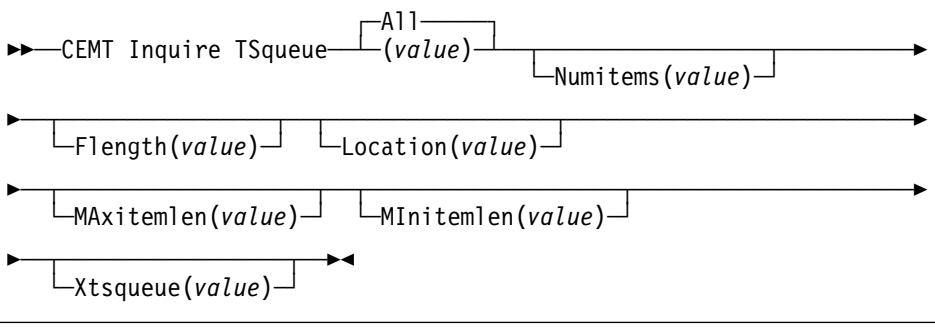
The command details are as follows:

CEMT INQUIRE TSQUEUE

Function

Retrieve information about temporary storage queues.

Syntax



Parameters

All is the default.

Tsqueue
indicates that this panel relates to a TSQUEUE inquiry.

(value)
displays the 8-character name of a temporary storage queue.

Numitems**(value)**
displays the number of items in the temporary storage queue.

Flength**(value)**
displays the total length in bytes of all the items in the temporary storage queue. For information about how CICS calculates the length of items, see the *CICS/ESA CICS-Supplied Transactions*.

Location**(value)**
displays where the temporary storage queue resides. The values are:

Auxiliary
The temporary storage queue is held on the CICS temporary storage VSAM data set, DFHTEMP.

Main
The temporary storage queue is held in main storage.

Maxitemlen**(value)**
displays the length in bytes of the largest item in the temporary storage queue. For information about how CICS calculates the length of items, see the *CICS/ESA CICS-Supplied Transactions*.

Minitemlen**(value)**
displays the length in bytes of the smallest item in the temporary storage queue. For information about how CICS calculates the length of items, see the *CICS/ESA CICS-Supplied Transactions*.

Xtsqueue**(value)**
displays the name of the temporary storage queue in hexadecimal representation.

Warning message for migrated data sets

This section outlines the change in CICS/ESA 4.1 to improve CICS support of SMS-managed user VSAM data sets. These changes are to enable you to recognise that an SMS-managed VSAM data set has been migrated to primary or secondary storage, and that opening the file will be delayed correspondingly.

These changes, outlined below, are for VSAM data sets managed by CICS file control under DFHSMS.

If you use DFHSMS to manage your VSAM data sets, you should consider carefully the period after which your CICS VSAM data sets are migrated to primary or secondary storage. If a migrated data set has to be recalled for CICS, it can take several minutes from primary storage, or longer from secondary storage. While the recall is taking place, the user transaction is suspended, and no other opens on that data set can be performed until the data set is recalled.

If a migrated data set has to be recalled, CICS issues message DFHFC0989 to the system console, to notify the operator that a recall is taking place, and to indicate whether it is from primary or secondary storage.

For details of all the new, changed, and obsolete messages in CICS/ESA 4.1, see the *CICS/ESA Migration Guide*.

Shutdown-assist sample program DFH$SDAP

The shutdown-assist sample program, called DFH$SDAP, is designed to run in the first phase of CICS shutdown to resolve any problems that are preventing CICS perform a successful warm shutdown with a keypoint.

This sample program is an assembler-language program, provided in CICS410.SDFHSAMP. It can be invoked during first phase of shutdown by means of an entry in a shutdown program list table (PLTSD). It is designed to ensure that, during normal shutdown, CICS is not suspended for any significant length of time while waiting for an event that might never happen. It aims to purge user tasks, with inflight tasks being backed out and a warm keypoint taken.

The program has several phases, whereby it progressively attempts to remove tasks that are inhibiting a successful warm shutdown. The phases are as follows:

1. The sample program attempts to remove user tasks by the EXEC CICS SET TASK(...) PURGE command.
2. If the PURGE command fails to remove all user tasks, the program goes on to the next phase, which is to us EXEC CICS SET TASK(...) FORCEPURGE command.
3. If FORCEPURGE fails to remove all user tasks, the program goes on to issue an EXEC CICS SET VTAM FORCECLOSE command.
4. Finally, if the previous phases have not succeeded, the program overrides the normal shutdown with an EXEC CICS PERFORM SHUTDOWN immediate.

The program restarts itself between each phase, using an EXEC CICS START command to restart transaction SDAP with an interval arbitrarily set to 10 seconds.

Chapter 33. Changed and new utility programs

This chapter outlines changes in CICS/ESA 4.1 to CICS-supplied utility programs, including new utility programs. The changes described here are to the:

- Message editing utility program. This is a new program.
- In-doubt window resolution utility program, DFH$IWUP. This is a new program.
- IPCS SDUMP formatting program.
- Monitoring utility program, DFHMNDUP.
- Statistics utility program, DFHSTUP.
- Trace utility program, DFHTU410. This was known as DFHTUP in earlier releases.

Other changes to the utility programs are:

- The DFHDUP transaction dump utility program has been renamed DFHDU410
- The DFHTRGTF GTF trace utility program has been renamed DFHTR410.

Message editing utility program

In CICS/ESA 4.1 CICS provides a message editing utility to enable you to edit the CICS message tables.

You can use the message editing utility to change the text or language of CICS messages, and reassemble the message modules for use by your CICS regions.

Requirements

To be able to use the message editing utility, you need the following:

DASD space

The message editing utility needs 2.5MB for the programs and panels, and 9MB for the source English message dataset. The utility allocates 4MB for each target language.

ISPF Version 3

The message editing utility requires a minimum ISPF level of Version 3.

Access authority

To use the message editing utility, you need alter authority for the target data sets index, defined in “Defining the utility data set index” on page 538.

Installing the message editing utility

The library data sets and modules needed by the message editing utility are installed as part of the CICS/ESA 4.1 installation process. (The utility data sets are outlined in “Utility data sets” on page 538.) However, before you can use the message editing utility, you must define the ISPF index of the utility data sets. This is described in “Defining the utility data set index” on page 538.

Utility data sets

The following partitioned data sets are used by the message editing utility.

Message source data set
This data set, CICS410.SDFHSRCE, contains all English message source (DFHMExxE) files.

Executable files (CLISTs) data set
This data set, CICS410.SDFHCLIB, contains the message editing utility executable CLIST.

Load library
This data set, CICS410.SDFHLLIB, contains the load modules for the message editing utility.

Messages library
This data set, CICS410.SDFHMLIB, contains the modules for messages internal to the message editing utility.

ISPF panel library
This data set, CICS410.SDFHPLIB, contains the panels for the message editing utility.

Internal tables library
This data set, CICS410.SDFHTLIB, contains the utility-generated tables to control the tracking and processing of message data set members.

Input table of CICS language codes
This data set, CICS410.SDFHLANG, contains the table of all valid language codes supported by CICS.

These data sets, except CICS410.SDFHTLIB, are created automatically when you install CICS/ESA 4.1. The CICS410.SDFHTLIB data set, and some control files, are created automatically when you run the message editing utility. The control files are called target_dataset_index.MEUCNTLx, where:

target_dataset_index
is the index for all message editing utility target data sets.

x
is the CICS one-character language code.

Defining the utility data set index

If, when installing CICS, you change the location of the initial dialog module from CICS410.SDFHCLIB(DFHMEUCL), you must edit DFHMEUCL and change the PROC statement. The PROC statement specifies the location identifier of the utility, and is used to find the initial program for the message editing utility. As supplied, the PROC statement is as follows:

```
PROC 0 MEULIB(CICS410.DFHMEU)
```

If you want to invoke the message editing utility with a different dataset prefix, you can pass the dataset name from the command table. Alternatively, you can use the MEULIB(xxxxxxxx.xxxxxxxx) parameter on the CLIST command, where xxxxxxxx.xxxxxxxx is the prefix that you want to use. For example:

```
TSO EX 'CICS410.SDFHCLIB(DFHMEUCL)' 'MEULIB(mymeu.prefix)'
```

where *mymeu.prefix* is the prefix to be used for the utility data sets. MEULIB need only be specified if the prefix has been changed from the default.

Using the message editing utility

The process for running the message editing utility is generally:

Step	Task	For details, see page
1	Start the message editing utility	539
2	Specify your message editing utility default values (the first time run)	540
3	Perform actions on message data sets (from main panel) such as: • Copy message data set members • Select message sets to be edited • Edit selected messages • Assemble and link-edit changed message set members • Generate a message load module	543
4	Add the new message module to STEPLIB. For each CICS region that is to use the new message module, specify the corresponding language code on the NATLANG system initialization parameter.	547
5	(If needed) Apply PTFs	548

These tasks are described in the following sections, as identified.

Restriction on running the message editing utility: For each target data set index only one user can access the same selected language at a time. However, several users can have the same target data set index for different languages.

A TSO user must not use the message editing utility concurrently from two split screen sessions.

1. Starting the message editing utility

You can start the message editing utility by one of the following methods:

- Add MEU as an option to an ISPF menu.
- Add an entry to the ISPCMDS table. For example:

  ```
  MEU    3 SELECT CMD(EXEC 'CICS410.SDFHCLIB(DFHMEUCL)' 'MEULIB(CICS410)')
  ```

- Execute as a TSO function from the ISPF command line. For example:

  ```
  TSO EX 'CICS410.SDFHCLIB(DFHMEUCL)' 'MEULIB(CICS410)'
  ```

Note: If you are starting the message editing utility after it has failed, you must first delete the control files from the previous run before the utility can be restarted. The control files are called target_dataset_index.MEUCNTLx, where:

target_dataset_index is the index for all message editing utility target data sets.

x is the CICS one-character language code.

For more information about using ISPF dialog services to start functions (such as the message editing utility), see the *ISPF Dialog Management Guide and Reference*.

2. Specifying default values for the message editing utility

When you start the message editing utility for the first time, the *Set defaults* panel 1 (of 2) is displayed for you to enter your default values for the utility. For example, see Figure 42.

Note: When you first start the message editing utility, the following message is overlaid on the CICS macro library and CICS SDFHAUTH library lines; but after you press the ENTER key, the messages is removed.

```
MEU017  Defaults must be set before the Message Editing Utility can be used.
```

While entering the defaults you can press the ENTER key to save the values as you progress. When you have entered all the required default values, you can save the values and exit the panel by using End (F3). This either returns you to the “Message Editing Utility” main panel or displays the *Set defaults* panel 2, as shown in Figure 43 on page 541.

When you have defined your default values, any subsequent start of the message editing utility displays the message editing utility *Main* panel first.

```
                  Message Editing Utility - Set defaults          Page 1 of 2
                                                                 More:   +

 Values are saved when ENTER, Forward (F8), or End (F3) is pressed.

 MEU target data sets index  userid.MEU__________

 Current language (NATLANG)  A  +   (ENG) Alternative English

 Replace members during
     English source copy?  . NO_   Yes/No

 SMP/E maintained SDFHSRCE . CICS410.SDFHSRCE_____________________________

 DFHMEUL load library  . . . userid.load.library_____________________________

 CICS macro library  . . . . CICS410.SDFHMAC_____________________________

 CICS SDFHAUTH library . . . CICS410.SDFHAUTH_____________________________
COMMAND ===> ____________________________________________________________
 F1=Help      F2=Split     F3=End       F4=Language  F5=Refresh   F8=Forward
 F9=Swap     F12=Cancel
```

Figure 42. Message editing utility set defaults panel (1 of 2)

Warning

After the *MEU target data sets index* and the *SMP/E maintained SDFHSRCE* data sets have been set they should not be changed. The SDFHSRCE data set is used to create the source data set used internally by the message editing utility, based on the data set name specified as the target data set index. These data sets are also used as a base for the PTF update job. Changing either can result in inconsistencies in the message files.

The **+** sign beside the current language suffix field indicates that further help is available. Select Language (F4) to view the *Language selection* panel. The language suffix that you select is shown in the *Current language (NATLANG)* field.

To refresh the values back to the values last saved, use Refresh (F5).

You are recommended to use one target data set index for all languages. This makes it easier to create message modules for all languages, and to apply any PTF updates for the utility.

To create a message module, the utility needs to find all the translated messages for a language under the same target data set index. Therefore, you should not split the messages modules for a language between data sets with different indexes.

When the utility creates a new message module, it adds the module to the DFHMEUL load library specified on the *Set defaults* panel. For CICS to use this library, it must be APF-authorized, and added to the STEPLIB concatenation of your CICS startup job. (Alternatively, you can copy the new message module to another library in the STEPLIB concatenation.)

The CICS SDFHAUTH library specified on the *Set defaults* panel is used only to find the message module for messages that have not been edited.

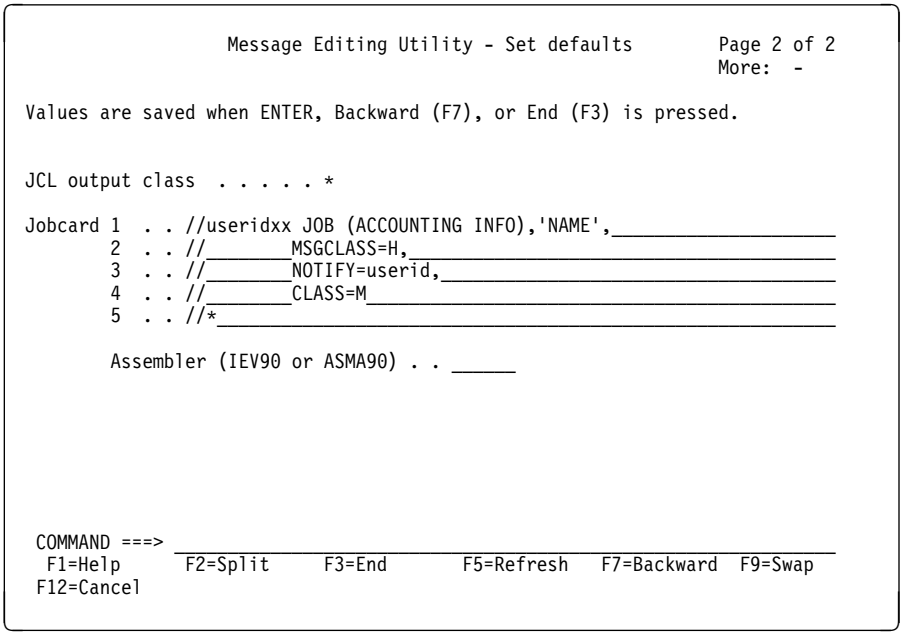


Figure 43. Message editing utility set defaults panel (2 of 2). This defines the job statement information added to the JCL to link-edit and generate message load modules, and to apply PTF updates. It also defines which assembler will be used in the link-edit JCL.

APAR PN79349

Documentation for PN79349 added on 25 April 1996

You can specify, on the “Message Editing Utility - Set defaults (Page 2 of 2)” shown in Figure 43, which assembler the utility is to use in the link-edit JCL. If you intend to use the High Level Assembler, specify ASMA90. Otherwise, specify IEV90. These are the only two values that are accepted.

Selecting languages for message translation: You can select the language to be used in your message set members on the *Language selection* panel. An example of the *Language selection* panel is shown in Figure 44 on page 543. The languages supported by the message editing utility are listed in the *Language selection panel* and for reference in Table 55 on page 542.

Table 55. Languages and codes supported by the message editing utility

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. The following language module suffixes are not supported by the message editing utility:

   E - US English master data sets
   K - Japanese data sets, where translation is performed by IBM.
3. 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.
4. You can select only one language a particular message editing utility edit session.
5. The NATLANG code for the selected language is used as the suffix of your edited message data sets to be created from the English language message data sets.

```
                    Language selection    ROW 1 TO 10 OF 33

 Use / to select a language, then press ENTER.

  NATLANG  Status  NLS   Language                      DBCS
                   code
_    A     Copied  ENG   Alternative English
_    B             PTB   Brazilian Portuguese
_    C             CHS   Simplified Chinese           DBCS
_    D             DAN   Danish
_    F             FRA   French
_    G             DEU   German
_    H             KOR   Korean                       DBCS
_    I             ITA   Italian
_    J             ISL   Icelandic
_    L             BGR   Bulgarian

COMMAND ===>_____________________________SCROLL ===> PAGE
 F1=Help        F2=Split       F3=End         F7=Backward
 F8=Forward     F9=Swap       F12=Cancel
```

Figure 44. Message editing utility language selection panel

Languages that have already been set up and used are indicated by the status *Copied*.

To select a language type a ***/*** character in the field to the left of the NLS code column and press ENTER.

3. Performing actions on message data sets

You can select message sets to be changed from the *Main* panel of the message editing utility. An example of the *Main* panel is shown in Figure 45 on page 544.

The *Main* panel provides for:

- Copying message data set members
- Selecting message sets to be edited
- Assembling and link-editing changed message set members
- Generating a message load module
- Sorting the list of message set members.

```
                    Message Editing Utility - Main panel    ROW 1 TO 17 OF 73

  Current language: Alternative English
  MEU index: userid.MEU

  Type one or more action codes.  Then press ENTER.

  C Copy  E Edit  L Link-edit                    Sort sequence: English name
                                                 Last change
Action  English name  New name  Status      Date     Time     Userid     Size
 _      DFHMEACE      DFHMEACA  .           1994/01/20 09:54  userid      1205
 _      DFHMEAIE      .         .           .                 .          .
 _      DFHMEAKE      .         .           .                 .          .
 _      DFHMEAME      .         .           .                 .          .
 _      DFHMEAPE      DFHMEAPA  Edited      1994/01/20 10:54  userid       205
 _      DFHMEBPE      .         .           .                 .          .
 _      DFHMECCE      .         .           .                 .          .
 _      DFHMECEE      .         .           .                 .          .
 _      DFHMECPE      .         .           .                 .          .
 _      DFHMECRE      .         .           .                 .          .
 _      DFHMEDBE      .         .           .                 .          .
 _      DFHMEDDE      .         .           .                 .          .
 _      DFHMEDEE      .         .           .                 .          .
                                                                         .
COMMAND ===> _____________________________________________ SCROLL === PAGE_
 F1=Help      F2=Split     F3=End       F5=Generate  F6=Sort      F7=Backward
 F8=Forward   F9=Swap     F10=ApplyPTF F11=Defaults F12=Cancel
```

Figure 45. Message editing utility main panel

All the current English message source members are displayed on the *Main* panel. Use Forward (F8) and Backward (F7) to scroll up and down the list.

The status shown for a member is always the last action performed on that member. For example, if the action performed was **Copy** and a previously copied version of the source exists, the status changes to one of the following:

- **Replaced**, if you have set the default parameter *Replace members during English source copy?* to Yes. The English source member is copied to your target source data set.
- **No-Replace**, if the value of the default parameter *Replace members during English source copy?* is No. The English source member is *not* copied to your target source data set. If you wanted to copy the source member, you can change the default and repeat the copy command.

The *Set defaults* panel can be accessed by Defaults (F11). (See Figure 42 on page 540.)

The *PTF update* panel can be accessed by ApplyPTF (F10). (See Figure 48 on page 548.)

The other actions that you can perform from this panel are described in the following sections:

Copying message data set members: To create your own language source member for an English message source member, type C against the member name, then press ENTER. This copies the English source member to your language source data set as a member with the suffix of the current default language (as specified on the *Set defaults* panel). For example, with the default language as Spanish, the member DFHMEACE is copied from the CICS410.SDFHSRCE data set to the member DFHMEACS in the

| your_meu_index.SDFHSRCS data set. If the target member exists and the replace
| option on the *Set defaults* panel has been selected, the member in the target data
| set is replaced and the status changes to Replaced. Otherwise, the status changes
| to No-Replace.

| ***Selecting message sets to be edited:*** To edit messages in a source member,
| type E against the member name, then press ENTER. This displays the *Message
| number selection* panel, which lists the messages in the source member and
| enables you to select messages to be changed. If the selected source member
| has not previously been copied, requesting the edit action copies the source
| member before editing it.

| **Notes:**

| When editing RP messages, these messages are split between four message
| sets as follows:

- DFHMEROx for message numbers from 0000 to 0700
- DFHMERPx for message numbers from 0701 to 0850
- DFHMERQx for message numbers from 0851 to 5200
- DFHMERRx for message numbers from 5201 to 9999

| When editing ZC messages, these messages are split between four message
| sets as follows:

- DFHMEZAx for message numbers from 0000 to 2099
- DFHMEZBx for message numbers from 2100 to 3399
- DFHMEZCx for message numbers from 3400 to 5899
- DFHMEZDx for message numbers from 5900 to 9999

| When you select a message set to be edited, the utility scans the file for the
| message numbers that it contains, and displays the *Message number selection*
| panel for those messages. The message numbers displayed are all the
| translatable messages from that message set. Any messages that cannot be
| translated are identified as such by a note added to the message in the *CICS/ESA
| Messages and Codes*. An example of the *Message number selection* panel is
| shown in Figure 47 on page 547.

| To select a message to be edited, type a / character in the field to the left of the
| message number, then press ENTER. When you press ENTER the *Edit message*
| panel (see Figure 46 on page 546) is displayed for the messages selected.

| ***Editing selected message sets:*** You can use the *Edit message* panel to change
| the text and reply inserts, and the order of inserts, of selected messages. An
| example of the *Edit message* panel is shown in Figure 46 on page 546.

```
                    Message Editing Utility - Edit message

   Message number: AC2035

 ****** **************************** TOP OF DATA ****************************
 419000 text " An invalid error code has been passed "
 420000 text "to DFHACP. "
 421000 text " Transaction "
 422000 special_insert tranid
 423000 text " is terminated. "
 424000 text " Terminal "
 425000 special_insert termid
 426000 text "."
 ****** ************************** BOTTOM OF DATA **************************



COMMAND ===> _____________________________________________ SCROLL ===> PAGE_
 F1=Help      F2=Split     F3=End       F4=Refresh   F5=Rfind     F6=Rchange
 F7=Backward  F8=Forward   F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 46. Example of message editing utility edit message panel

See “Rules for editing and translating” on page 550 for information and rules about editing CICS messages.

Assembling and link-editing the changed message data sets: Before you link-edit a source message member, ensure that it is not being used, because this prevents the link-edit job from running.

To link-edit a message source member, type L against the member name, then press ENTER. This submits a job to JES to convert the message source member into assembler language.

Note: Check the link-edit job output to ensure that it completed successfully. If not, examine the error messages generated, correct the message set and re-submit the link-edit job.

Generating a message load module: When you have edited and link-edited all the messages source members that you require, the next step is to create a load module to use with your CICS regions. To do this, use Generate (F5) on the *Main* panel. This assembles the link-edited version of all your message members with the English version of any you have chosen not to translate. Successful completion of this step results in DFHMET1x and DFHMET5x load modules being placed in the data set specified in the DFHMEUL load library field on the *Set defaults* panel. These modules can be used with your CICS job by placing them in the relevant APF-authorized library and specifying the associated language code on the NATLANG system initialization parameter.

Sorting the lists of message set members: To sort the lists of message sets displayed on the *Main* panel use the sort function key (F6). You can sort the message set member list by:

- English name
- New name
- Status
- Date and time.

Each time you press the sort function key the next sort order in the above list is selected.

4. Add the new message load modules to STEPLIB

To enable your CICS region to use the message load module generated by the message editing utility, you must:

- Add the module to a library in the STEPLIB concatenation of your CICS startup JCL.
- Specify the language character suffix of the module on the NATLANG system initialization parameter for your CICS startup job.

Note: CICS always loads the standard English message table by default, regardless of what you sepcify on the NATLANG system initialization parameter. To ensure your own message tables are selected as the default tables, specify your own language code first on the NATLANG parameter.

Examples: If you modify messages using language code A (for alternative ENGLISH) you should specify NATLANG=A (or NATLANG=(A,E) to ensure your modified message tables are used as the default tables in place of the standard English versions. NATLANG=A is equivalent to NATLANG=(A,E). Do **not** specify (E,A).

If you translate messages using S (for Spanish) and F (for French) and you want French to be the default language with Spanish and English used selectively (by terminal or userid), specify NATLANG=(F,S) or NATLANG=(F,S,E). In this example, if NATLANG is not specified on terminals or userids, French is taken as the default language.

```
        Message Editing Utility - Message number selection  ROW 1 TO 11 OF 11

  Message module: DFHMEACA

  Use / to select one or more messages and press ENTER.

  _   2001    _   2002    _   2003    _   2004    _   2005    _   2006
  _   2007    _   2008    _   2009    _   2010    _   2012    _   2014
  _   2015    _   2016    _   2017    _   2018    _   2019    _   2020
  _   2021    _   2022    _   2023    _   2024    _   2025    _   2026
  _   2027    _   2028    _   2029    _   2030    _   2033    _   2034
  /   2035    _   2036    _   2037    _   2038    _   2039    _   2040
  _   2041    _   2042    _   2043    _   2044    _   2047    _   2050
  _   2051    _   2052    _   2053    _   2054    _   2055    _   2056
  _   2057    _   2206    _   2207    _   2208    _   2230    _   2236
  _   2237    _   2238    _   2259    _   2260    _   2261    _   2262
  _   2263    _   2603    _   2605    _   2606    _           _
 ****************************** BOTTOM OF DATA ******************************



 COMMAND ===> _____________________________________________________ SCROLL == PAGE_
  F1=Help      F2=Split     F3=End       F7=Backward  F8=Forward   F9=Swap
 F12=Cancel
```

Figure 47. Message editing utility message number selection panel

5. Applying PTFs to the message editing utility

This section outlines the process that you use to apply service to the message data sets built via the message editing utility. This is necessary to keep the message files created by the message editing utility in step with the PTF level for your CICS system. It is important that the PTF update process is run whenever you update your CICS PTF level. Failure to do this can result in errors when your running CICS regions issue messages.

To apply PTF updates to your message files:

1. Apply PTFs to the SMP/E-maintained English source. This happens as part of the normal process of applying PTFs to CICS.
2. Select the ApplyPTF (F10) option of the *Main* panel. This displays the *Submit PTF update job* panel is displayed; for example, see Figure 48. To apply a PTF:
   a. Complete the data set details for the SMP/E maintained SDFHSRCE data set and the update log.
   b. Press ENTER to validate the input fields.
   c. Press Submit (F5) to submit the PTF update job.
3. When the update is complete check the output log for messages requiring re-translation.
4. Translate any messages as needed, and re-run the link-edit and generate jobs. This creates a new message module to use with your CICS jobs.

```
          Message Editing Utility - Submit PTF update job     ROW 1 TO 2 OF 2

Change the values below, press ENTER to save or F5 to Submit.


SMP/E maintained SDFHSRCE. CICS410.SDFHSRCE_____________________________

PTF update log . . . . . your.PTFLOG_____________________________________

Write log in upper case? . NO_   Yes/No

The PTF updates will be applied to the following data sets.

Data set                                        Language
userid.MEU.SDFHSRCA                             Alternative English
userid.MEU.SDFHSRCM                             Macedonian
***************************** BOTTOM OF DATA ********************************




COMMAND ===> _____________________________________________________ SCROLL ===>____
 F1=Help      F2=Split     F3=End       F5=Submit    F7=Backward  F8=Forward
 F9=Swap     F12=Cancel
```

Figure 48. Message editing utility submit PTF update job panel

When the details have been completed and verified, press the Submit (F5) to instruct the message editing utility to build and submit a TSO CLIST to apply the PTF updates. This CLIST is intended for TSO Background execution only.

Immediately after pressing Submit (F5), the message editing utility terminates. The utility is prevented from restarting while the PTF update job is in progress. If the update process should fail for any reason, two datasets will be left over that will prevent the message editing utility from running. In this situation the following datasets can safely be deleted:

userid.MEU.PTFJOB
userid.MEU.PTFCLIST

The submit PTF update job can now be restarted.

Guidelines for PTF update job.

The PTF update job does the following processing:

- New messages are added to all the source datasets.
- Redundant messages are flagged as deleted by placing an ***** in column 1 of the message definition.
- Changed messages are refreshed with the English message for ***all*** languages. The old message details are written to the PTF update log.
- All operations are recorded in the PTF update log. (See sample log output in "PTF update log sample output.") Progress messages are output to the console while the CLIST is running.

PTF update log sample output

Figure 49 on page 550 shows a sample of the output for the message editing utility PTF update log.

```
DFHMEUU *** PTF update program started *** yy/mm/dd hh:mm:ss

DFHMEUU PTF COMPARISON STARTED
COMPARING PTF DFHMEACE - ENGLISH DFHMEACE
COMPARING PTF DFHMEAIE - ENGLISH DFHMEAIE
COMPARING PTF DFHMEAKE - ENGLISH DFHMEAKE

DFHMEUU PTF COMPARISON COMPLETED

DFHMEUU ---------------------------------------

DFHMEUU PTF UPDATE STARTED   userid.MEU.SDFHSRCE

DFHMEUU DFHMEAPE UPDATED MESSAGE 0701 - OLD DETAILS FOLLOW
SPECIAL_INSERT APPLID                                                 07300000
TEXT " AN ABEND (CODE "                                               07310000
INS#1 FORMAT CHAR PUBSCHAR "ABCODE"      * 6-CHAR ABEND CODE          07320000
TEXT ") HAS OCCURRED IN EXIT PROGRAM "                                07330000
INS#2 FORMAT CHAR PUBSCHAR "PROGNAME"    * 8-CHAR EXIT PROGRAM NAME   07340000
TEXT " AT EXIT POINT "
INS#3                                    * 8-CHAR EXIT POINT NAME
TEXT "."

DFHMEUU DFHMEDEE ADDED   MESSAGE 0118

DFHMEUU DFHMETDE ADDED   MESSAGE 1280

DFHMEUU DFHMETOE DELETED MESSAGE 6024

DFHMEUU PTF UPDATE COMPLETED userid.MEU.SDFHSRCE

DFHMEUU ---------------------------------------

DFHMEUU PTF UPDATE STARTED   userid.MEU.SDFHSRCA

DFHMEUU PTF UPDATE COMPLETED userid.MEU.SDFHSRCA

DFHMEUU *** PTF UPDATE PROGRAM COMPLETED *** yy/mm/dd hh:mm:ss
```

Figure 49. Message editing utility PTF update log sample output

Rules for editing and translating

When editing messages, you ***must*** observe the following rules.

Message items that must not be altered: You must not alter the following types of message item:

1. ins#n format (CHAR]HEX]DEC]TIME]DATE) pubschar "xxxxx"
2. special_insert xxxxxxxx

These types of inserts must not be changed in any way. However, when editing the message, you can alter the order and position of the inserts within the message, to make the sentence structure more appropriate, but must not change the insert number. The positioning of inserts in the message template determines the location of the inserts in the output message. The suffix #n associates the insert with a variable in CICS code; it does ***not*** denote its position in the output message. For example, the following message details in English:

```
"text...",ins#1,"text....",ins#2
```

in another language might be:

```
ins#2,"text...",ins#1,"text...."
```

```
                    Message Editing Utility - Edit message

   Message number: AC2016

 ****** **************************** TOP OF DATA ****************************
 228000 text " Transaction "
 229000 special_insert tranid
 230000 text " cannot run because program "
 231000 ins#1 format CHAR pubschar "program name"
 232000 text " is not available."
 ****** ***************************** BOTTOM OF DATA ******************


COMMAND ===> ____________________________________________ SCROLL ===> PAGE_
 F1=Help      F2=Split     F3=End       F4=Refresh   F5=Rfind     F6=Rchange
 F7=Backward  F8=Forward   F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 50. Message editing utility edit message panel showing types of inserts

In the example in Figure 50, line numbers 229000 and 231000 can be moved but must not be altered in anyway.

Message items that can be altered: The message editing utility limits the editing to the message text, to maintain the integrity of the message definition. You can alter the following types of message item:

1. text “text_string”
   ins#n format OPT value#n "text_string"

   You can translate the text, text_string, which appears between the two double quotes or text delimiters. The “text_string” must not extend beyond column 72 or be continued onto the next line. If more than one line is required for the text, another *text “text_string”* record must be added. The text may be in upper or mixed case. Double-byte text must be enclosed in shift-out and shift-in delimiters within the text_string.

   For optional inserts, OPT value#n, the value#n can spread over several adjacent lines. If you move such an insert, you must move all subsequent value#n lines that are part of the insert. If you do not move all value#n lines for an insert, the message editing utility does not detect this, but CICS will issue an error message if it tries to issued such an incompletely edited message.

   An example of this type of message is shown in Figure 51 on page 552. In this example, line numbers 625850 and 625870 must be moved together, and line 625870 must remain below line 625850.

```
                    Message Editing Utility - Edit message

   Message number: SI1502

 ****** **************************** TOP OF DATA ****************************
 625830 text " CICS startup is "
 625850 ins#1 format opt value#1 "Cold" value#2 "Warm"
 625870 value#3 "Emergency" value#4 "Logterm"
 625890 text "."
 ****** ***************************** BOTTOM OF DATA ***************************


COMMAND ===> _____________________________________________ SCROLL ===> PAGE_
 F1=Help      F2=Split     F3=End       F4=Refresh   F5=Rfind     F6=Rchange
 F7=Backward  F8=Forward   F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 51. Message editing utility edit message panel, opt insert split over lines

2. reply#n “text_string”

   These are a special form of message insert which also serve to define the reply values for a console message requiring an operator reply. They are not be applicable to DBCS languages, because console messages cannot be translated into DBCS languages, unless they are sent to a TDQ destination as well. The positional rules are the same as for other types of inserts. As with the value#n keyword, the text_string within the double quotes following the reply#n keyword may be translated. The text_string ***must*** be in upper case. An example of this is shown in Figure 52.

```
                    Message Editing Utility - Edit message

   Message number: AP0100

 ****** **************************** TOP OF DATA ****************************
 554200 text " Suffixed module "
 554800 ins#1                                format CHAR pubschar "modnane"
 555400 text " cannot be loaded. ENTER new suffix, "
 555600 reply#1 "YES"
 555800 text "(unsuffixed), "
 556000 reply#2 "NONE"
 600000 text "(dummy), or "
 640000 reply#3 "CANCEL"
 ****** ***************************** BOTTOM OF DATA ***************************


COMMAND ===> _____________________________________________ SCROLL ===> PAGE_
 F1=Help      F2=Split     F3=End       F4=Refresh   F5=Rfind     F6=Rchange
 F7=Backward  F8=Forward   F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 52. Message editing utility edit message panel showing reply#n over several lines

Note for DBCS languages

If a message has a destination of TERMCDBC or CONSOLE then it must not be translated into a DBCS language. If a message has a destination of CONSOLE and TDQ then it can be translated.

Overall message length: The different message destinations have different maximum message lengths. If these are exceeded the message will be truncated. The number of bytes specified for each destination is after the message identifier

and default leading inserts have been taken into account, all you need consider is the text you are presented in the *Edit message* panel.

- Console message: Converse messages (that is, those requiring a user response) must not exceed 95 bytes. Other console messages must not exceed 600 bytes.
- Transient data queue messages must not exceed 1200 bytes.

In calculating the overall message length, you must include the lengths of both inserts and text strings. The following is a guide to the lengths of inserts and special_inserts;

- Insert fields (depending on type)

```
CHAR       n bytes (specified in insert comment)
HEX        up to 14 bytes
DEC        up to 6  bytes
OPT        translatable field of variable length
```

- Special_inserts

```
APPLID            9 bytes
SYSID             5 bytes
DATE              9 bytes
TIME              9 bytes
TRANID            5 bytes
TERMID            5 bytes
PROGRAM_NAME      9 bytes
USERID            9 bytes
NETNAME           9 bytes
PRIMARY_ABCODE    5 bytes
SECONDARY_ABCODE  5 bytes
```

  All the special_inserts have a trailing blank which has been taken into account.

Change flags: Some lines have a symbol such as '@PA' at the end of the line. These symbols are IBM internal change flags and can be removed or over typed if needed.

Getting help

From any message editing utility panel, you can press Help (F1) to display help information relevant to that panel.

If you press Contents (F11) from any help panel, the message editing utility *Help contents* panel is displayed.

```
                    Message Editing Utility - Help contents


The Message Editing Utility provides a means of editing and translating
CICS messages.

Users new to this utility are advised to review the general information topic.
For further information refer to CICS/ESA Operations and Utilities Guide.

The following topics may be selected by number:


                         1 - General information
                         2 - Main panel
                         3 - Setting the system defaults
                         4 - Selecting a language suffix
                         5 - Selecting a message number
                         6 - Editing a message
                         7 - Submit PTF update job




COMMAND ===>
 F2=Split   F9=Swap   F12=Cancel
```

Figure 53. Message editing utility help contents panel

To display help information for a specific topic, type the number for the topic, press ENTER.

In-doubt window resolution utility program, DFH$IWUP

If a CICS region fails during a syncpointing process, other CICS regions with which it communicates do not know whether the failed CICS region has backed out or committed its updates. The period from the last syncpoint to the time of failure is known as the **in-doubt window**. For this period, the resource data is potentially inconsistent between the failed CICS region and other CICS regions that also use the same resources.

The danger is that after the failed CICS region has restarted, new transactions can use the inconsistent data; they may read or update the data, and in either case the results of these new transactions can be corrupted.

If a CICS region fails, it issues message DFHZN2101 to identify any units of work, relating to specific tasks, which are considered to be in-doubt. When the CICS region recovers from the failure, it issues further messages to resolve the in-doubt tasks to “In sync” or “Out of sync”. (The latter case is reported by message DFHZN2103.) The messages indicate that data integrity may be exposed, but they do not indicate which resources and records are exposed. Also, only CICS resources are reported; not DB2, DBCTL, or other resource types.

The in-doubt window resolution utility program, DFH$IWUP, helps you to resolve in-doubt resources by listing:

- The user data records, including the resource name and record key if appropriate, that were affected by in-doubt tasks.

This helps you to investigate the messages that report loss of integrity, enabling you to quantify the effect of the failure.

- The resource type, and type of request.

  This enables you to differentiate spurious messages from significant ones. (Spurious warning messages of in-doubt problems are issued by CICS regions that have not updated any resources.)

You should run the DFH$IWUP utility program against copies of the system logs for all CICS regions involved in the units of work that are in-doubt.

Overview of the in-doubt window resolution utility program, DFH$IWUP

The in-doubt window resolution utility program, DFH$IWUP, helps you to resolve in-doubt resource updates.

The DFH$IWUP utility program is an exit program to the CICS journal utility program, DFHJUP. It takes as input a list of unit-of-work IDs as they appear in the DFHZN210x messages generated to the CSMT log.

Note: You can copy the CSMT log and edit it to produce your own list of unit-of-work IDs to be investigated.

The DFH$IWUP utility program scans the journal, and outputs a report of all journal records for tasks relating to the unit-of-work IDs. The report lists, by task, for each journal record found:

- The resource type and type of request.

  This enables you to differentiate spurious messages from significant ones. (Spurious warning messages of in-doubt problems are issued by CICS regions that have not updated any resources.)

- The user data, including resource name and record key (where appropriate).

  This helps you to investigate the messages that report loss of integrity, enabling you to quantify the effect of the failure.

To resolve the in-doubt resource update operations, you must interpret the journal data to determine whether or not there has been a loss of integrity.

You can use the DFH$IWUP utility program to resolve:

- DFHZN2103 messages (for databases out of synchronization) after CICS has reported the result of its recovery logic
- Any units of work whose outcome is unknown after a recovery failure
- Operator action which has prevented CICS from resolving DFHZN2101 messages.

You should run the DFH$IWUP utility against copies of the system logs for all CICS regions involved in the units of work which are in-doubt.

Note: You cannot use the DFH$IWUP utility program to resynchronize resources.

Limitations of the DFH$IWUP utility program

You cannot use the DFH$IWUP utility program to resynchronize resources.

The DFH$IWUP utility program does not list unit-of-work IDs for CICS regions connected using LU6.1 protocols. That is, it is unable to resolve any in-doubt resources updated using LU6.1 protocols.

For more information about the DFH$IWUP utility program, see the *CICS/ESA Operations and Utilities Guide*.

Job control statements to assemble and link-edit the DFH$IWUP program

Before you can run the DFH$IWUP program, you must assemble and link-edit it into your CICS environment. You can use the CICS-supplied procedure, DFHASMVS, to assemble and link-edit the DFH$IWUP program. Figure 54 shows sample job control statements that you can use assemble and link-edit the DFH$IWUP utility program.

```
//IWUPASM JOB 'accounting information',CLASS=A,MSGCLASS=A,
//        REGION=2M
//*
//* INVOKE THE  DFHASMVS PROCEDURE TO ASSEMBLE DFH$IWUP
//*
//ASM    EXEC  DFHASMVS,
//             INDEX=CICS,
//             MOD=DFH$IWUP
//*
//*
//SYSPUNCH DD  DSN=&&OBJ,DISP=(,PASS),UNIT=SYSDA,
//             RECFM=FB,BLKSIZE=2960,LRECL=80,
//             SPACE=(CYL,(1,1))
//*
//SYSIN    DD  DSN=CICS.SAMPLIB(DFH$IWUP),DISP=SHR   /*IWUP SOURCE*/
//*
//* INVOKE THE  DFHLNKVS PROCEDURE TO LINK=EDIT DFH$IWUP
//*
//LKED   EXEC  DFHLNKVS,
//             PARM='LIST,LET,XREF',
//             INDEX=CICS,
//             INDEX2='your.prefix',   1
//             NAME=SDFHLOAD
//*
//*
//SYSPUNCH DD  DUMMY
//*
//SYSLIN   DD  DSN=&&TEMP,DISP=(OLD,DELETE)
//         DD  *
         MODE AMODE(24) RMODE(24)
         NAME DFH$IWUP(R)
/*
//
```

1 Change `your.prefix` to the prefix of your SDFHLOAD library to which the assembled and link-edited DFH$IWUP program is to be written.

Figure 54. Sample JCL to assemble and link-edit the DFH$IWUP utility program

Job control statements to run the DFH$IWUP program

To run the DFH$IWUP utility program, edit and submit the sample JCL shown in Figure 55.

```
//IWUPJOB JOB 'accounting information',CLASS=A,MSGCLASS=A,
//        REGION=2M
//*
//* INVOKE THE  JOURNAL UTILITY PROGRAM WITH EXIT DFH$IWUP
//*
//PRINT  EXEC  PGM=DFHJUP
//STEPLIB  DD  DISP=SHR,DSN=CICS.SDFHLOAD         /*CICS LIBRARY    */
//         DD  DISP=SHR,DSN=your.prefix.SDFHLOAD  /*DFH$IWUP LIBRARY*/
//SYSUT1   DD  DISP=SHR,DSN=CICS.J01A             /* JOURNAL IN     */
//UOWIDIN  DD  *                                  /* UOWID IN       */
llllllllllllllll.X'ttttttttttt'.nnnnn
llllllllllllllll.X'ttttttttttt'.nnnnn
/*
//SYSPRINT DD  SYSOUT=*,RECFM=FBA,LRECL=133,BLKSIZE=1330
//SYSIN    DD  *
OPTION   PRINT EXITR=DFH$IWUP
END
/*
//
```

Note: :
The JCL to run the DFH$IWUP utility program uses the following DD statements:

DDname Defines

STEPLIB
The CICS load library and, if different, the load library containing the DFH$IWUP module.

SYSUT1
The journal data set (in use at the time of failure) to be processed.

SYSPRINT
The output data set to which the formatted print records and control messages are to be sent by the DFH$IWUP utility program.

UOWIDIN
The input list of unit-of-work IDs to be scanned for. Each unit-of-work ID must be entered on a separate line, starting in column 1.
Each unit-of-work ID is of the format:

```
llllllllllllllll.X'ttttttttttt'.nnnnn
```

where

lllllllllllllll is the LU name

ttttttttttt is a hexadecimal token/clock value

nnnnn is the sequence number

Any file of unit-of-work IDs specified on the UOWID DD statement must have an 80-byte record format.

SYSIN
The input control data set for DFHJUP. This file must be in 80-byte record format.

Figure 55. Sample JCL to run the DFH$IWUP utility program

Creating your own DFH$IWUP utility program

You can create your own DFH$IWUP utility program, for which the source code is supplied in member DFH$IWUP of the CICS410.SDFHSAMP library.

IPCS SDUMP formatting program

In CICS/ESA 4.1, the following changes have been made to the IPCS SDUMP formatting program:

- You can select parts of the CICS internal trace table to format for a system dump. How you do this is described in "Selecting parts of the CICS internal trace table."
- The date and time at which the SDUMP was taken are displayed when the dump is formatted.

Selecting parts of the CICS internal trace table

You can select which parts of the CICS internal trace table to format for a system dump, by using a new CICS dump exit parameter, TRS, for the IPCS SDUMP formatting program: This parameter enables you to select trace entries by:

- Kernel task
- Task identifier
- Terminal
- Transaction identifier
- Time period
- Trace identifier.

To select the parts of the internal trace to be formatted by IPCS, you specify the TRS parameter on the IPCS VERBEXIT command, for example,

```
VERBEXIT CICS410 'DEF=1,DLI=1,KE=3,TR=2,TRS=<TRANID=CSSC,KE_NUM=12>'
```

Notes:

1. The VERBEXIT statement specifies the verb name CICS410 to process CICS/ESA 4.1 system dump data. This corresponds to the IPCS dump exit routine DFHPD410, as specified in the DFHIPCSP member in the CICS410.SDFHPARM library.
2. For the TRS parameter to work, you must also specify the TR parameter, without a value of 0, to use output from the trace domain.

Monitoring utility program, DFHMNDUP

The following new control parameters have been added to the dictionary utility program, DFHMNDUP:

JOBDATE=yyddd

Code this with the Julian date, as five numeric characters, of the MVS job to be included in the dictionary record.

yy represents the year (for example, 93 for 1993)

ddd represents the day, in the range 1 through 366.

If you do not specify a date, the current date is used.

JOBNAME=xxxxxxxx
Code this with an MVS job name for the CICS region to be included in the dictionary record.

JOBTIME=hhmmss
Code this with a time stamp, as six numeric characters, for MVS job to be included in the dictionary record.

hh the number of hours, in the range 00 through 24.

mm
the number of minutes, in the range 00 through 59.

ss the number of seconds, in the range 00 through 59.

If you do not specify a time, the current time is used.

USERID=xxxxxxxx
Code this with eight alphanumeric characters that represent the user identification of the MVS job to be included in the dictionary record. The user identification value *xxxxxxxx*, must correspond to any values that you have set up in your MVS IEFUSI exit, but does not have to be a real userid. For more information on the MVS job step initiation exit IEFUSI, see the &esaspluec. manual.

Statistics formatting utility program, DFHSTUP

The statistics utility program, DFHSTUP, has been enhanced to provide clearer layout and resource type selectivity. As with monitoring, there have been changes in the way the statistics are reported and correlated with other SMF data so that your tools can reflect the information gathered from your whole system. DFHSTUP also has:

- a new DD statement required to run DFHSTUP:

  ```
  //DFHSTWRK DD UNIT=SYSDA,SPACE=(CYL,(8,4))
  ```

- two new control parameters:

  SELECT TYPE
  IGNORE TYPE

  You can use these parameters to reduce the amount of statistics data processed by the statistics utility program, DFHSTUP, by specifying the types of statistics that you are interested in or want to ignore.

  The new control parameters are described in “Control parameters of the DFHSTUP program.”

Control parameters of the DFHSTUP program

SELECT TYPE={type|(type1[,type2]...[,typeN])}
Code this parameter with the resource types for which you want statistics to be formatted and printed. The parameter keywords must be coded as shown, with one blank between the two words. Code only one SELECT TYPE parameter or one IGNORE TYPE parameter. If you specify two or more resource types,

you must enclose them in parentheses, and separate them by commas. (The resource types that you can code on this parameter are listed in Table 56.)

If you do not code this parameter, the DFHSTUP program reports statistics for all resource types found in the DFHSTATS data set, other than those resource types specified on an IGNORE TYPE parameter.

IGNORE TYPE={type|(type1[,type2]...[,typeN])}

Code this parameter with the resource types for which you want the statistics ignored. The parameter keywords must be coded as shown, with one blank between the two keywords. Code only one IGNORE TYPE parameter. If you specify two or more resource types, you must enclose them in parentheses, and separate them by commas. (The resource types that you can code on this parameter are listed in Table 56.)

Code only one SELECT TYPE parameter or one IGNORE TYPE parameter.

If you do not code this parameter, the DFHSTUP program reports statistics for the resource types found in the DFHSTATS data set, depending on the SELECT TYPE parameter.

Table 56. DFHSTUP: Resource types that can be selected or ignored

AUTOINSTALL
CONNECTION
DBCTL
DCE
DISPATCHER
DLI
DTB
FEPI
FILE
IRCBATCH
JOURNAL
LSRPOOL
MONITOR
PROGAUTO
PROGRAM
STATS
STORAGE
SYSDUMP
TABLEMGR
TRANCLASS or TCLASS
TDQUEUE
TERMINAL
TRANDUMP
TRANSACTION
TSQUEUE
USER
VTAM

Trace utility print programs

This section describes how CICS/ESA 4.1 extends and improves the selectivity of trace entries in auxiliary trace and generalized trace.

It covers the following topics:

- Overview
- Benefits
- The trace control parameters.

Overview

You can select, format, and print trace entries from the A or B auxiliary trace data set by using the CICS utility program, DFHTU410. You control the selection of trace entries for formatting and printing by the program on trace control statements supplied in either:

- A PARM parameter on the EXEC PGM=DFHTU410 statement, or
- The DFHAXPRM data set.

You can specify that all entries are to be processed, or select entries for processing, such as entries:

- Written to the auxiliary trace data set within a specified period of time
- Written for a specified terminal
- With a specified trace identifier
- Associated with a specified transaction identifier
- Associated with a specified task
- Associated with a selected kernel task

You can select which trace entries you want to highlight in your formatted output by specifying:

- The time interval between one trace entry and the next being written.

  If more than the specified interval elapses before the next trace entry is written, then this next trace entry is formatted and printed with an asterisk (*) to draw your attention to this entry.

In CICS/ESA 4.1, the selectivity of trace entries to be output has been extended. In addition to selection methods already available, you can select trace entries by specifying:

- Trace entry sequence numbers (auxiliary trace and generalized trace facility).

  The sequence number is given in each trace entry, and can be determined from a summary trace print.

- Exception trace entries only (auxiliary trace only).

Benefits

The time interval between trace entries can be very useful in debugging problems. This is particularly the case where the problem appears to be performance-related.

By specifying the sequence numbers of required trace entries, you can select a much smaller number of consecutive trace entries than was possible using the TIMERG parameter in releases before CICS/ESA 4.1 (because the minimum time

range is two seconds and, in a busy CICS region, many hundreds of trace entries may be written in that time).

Being able to select only exception trace entries from an auxiliary trace enables you to narrow down the problem area before printing only those trace entries of interest (for example, for a particular time range, transaction, or sequence number). This helps when debugging intermittent problems, for which auxiliary trace may be used for extended periods. In releases before CICS/ESA 4.1, you would have to print the whole of a very large auxiliary trace in order to locate exception trace entries.

The trace control parameters

You can select such trace entries by specifying the following new control parameters:

ENTRY_NUM
EXCEPTION
INTERVAL

You can specify the ENTRY_NUM control parameter on either of the following:

- The DFHAXPRM DD statement or PARM statement of the DFHTU410 utility program
- The CICS parameter of the IPCS GTFTRACE subcommand.

You can specify the EXCEPTION and INTERVAL control parameters on the DFHAXPRM DD or PARM statement of the DFHTU410 utility program only.

The new trace control statements to define which trace records you want to print are:

ENTRY_NUM=({nnnnnn|nnnnnn-nnnnnn}[,{nnnnnn|nnnnnn},.,.,.])
This statement specifies the sequence numbers of one or more trace entries that you want to print. Each sequence number can be up to six digits in length. If you specify a range of sequence numbers, by using xxxxxx-yyyyyy, the second sequence number (yyyyyy) must be larger than the first (xxxxxx).

EXCEPTION
This statement specifies that only exception trace entries in the auxiliary trace data set are to be printed.

Note: This parameter is not valid for printing GTF trace entries.

INTERVAL={00.0128|number of seconds}
This parameter specifies the interval between auxiliary trace entries after which entries are highlighted with an asterisk, as follows:

- In abbreviated trace format, the asterisk appears to the left of the sequence number.
- In full trace format, the asterisk appears (as it does in releases prior to CICS/ESA 4.1 where a system–imposed time interval of 00.0128 seconds applies) as the next character after the printed time interval.

If successive auxiliary trace entries are written at intervals equal or greater than this limit, they are highlighted in the same manner.

If successive auxiliary trace entries are written at intervals less than this limit, they are not highlighted. They are, however, written, formatted and printed.

If you specify no INTERVAL value, a default of 00.0128 seconds applies.

You can specify interval values in the range zero seconds (where all trace entries would be highlighted) through 99.9999999999 seconds.

Note: The interval extends to ten decimal places. Zeros are padded from the right.

For more information about the CICS trace utility print programs, see the *CICS/ESA Operations and Utilities Guide*.

Part 8. Consolidated changes to externals

This Part shows the complete changes to the syntax of CICS API and SPI commands, to CICS supplied transactions, and to resource definition online. It also describes the hardware and software requirements for CICS/ESA 4.1

It includes the following topics:

Chapter 34. Changes to the application programming interface (API)

This chapter shows all the changes to the application programming interface made in CICS/ESA 4.1. The syntax diagrams show all the parameters, with the changed or new parameters marked with a vertical line to the left.

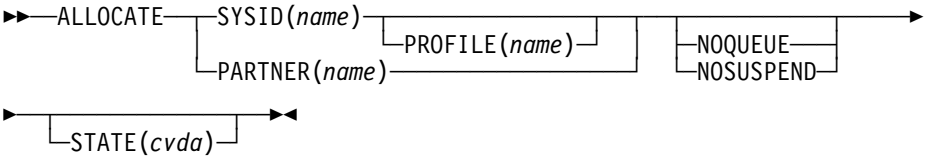
This chapter contains General-use Programming Interface information.

ALLOCATE(APPC)

Function

Acquire a session to a remote APPC logical unit for use by an APPC mapped conversation.

Syntax



Conditions:
CBIDERR, INVREQ, PARTNERIDERR, SYSBUSY, SYSIDERR

For details of the options PARTNER, PROFILE, and SYSID see “EXEC CICS ALLOCATE (APPC mapped)” on page 468.

ASSIGN

Function

Request values from outside the application program's local environment.

Syntax

Conditions:
INVREQ

For details of the options ASRAKEY, ASRASPC and ASRASTG see “EXEC CICS ASSIGN command” on page 26.

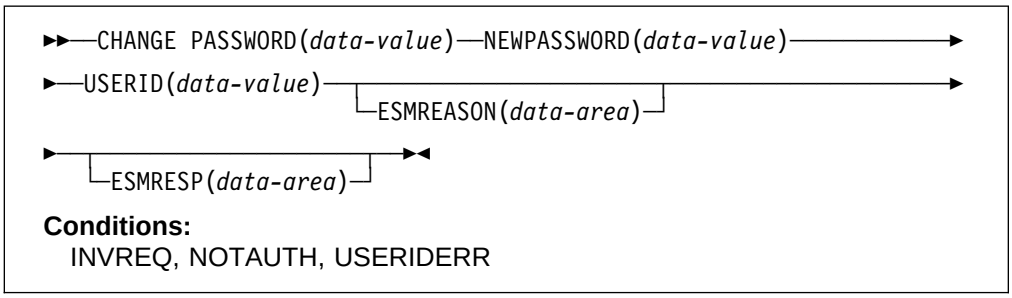
For details of the options INVOKINGPROG and RETURNPROG see “EXEC CICS ASSIGN command” on page 332.

CHANGE PASSWORD

Function

Change the password for a specified userid.

Syntax



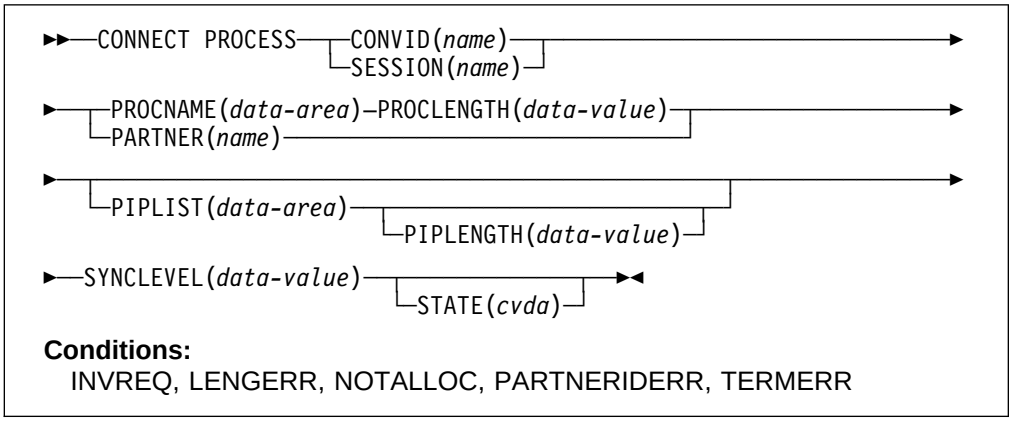
For details of the keywords see “EXEC CICS CHANGE PASSWORD command” on page 394.

CONNECT PROCESS

Function

Initiate an APPC mapped conversation.

Syntax



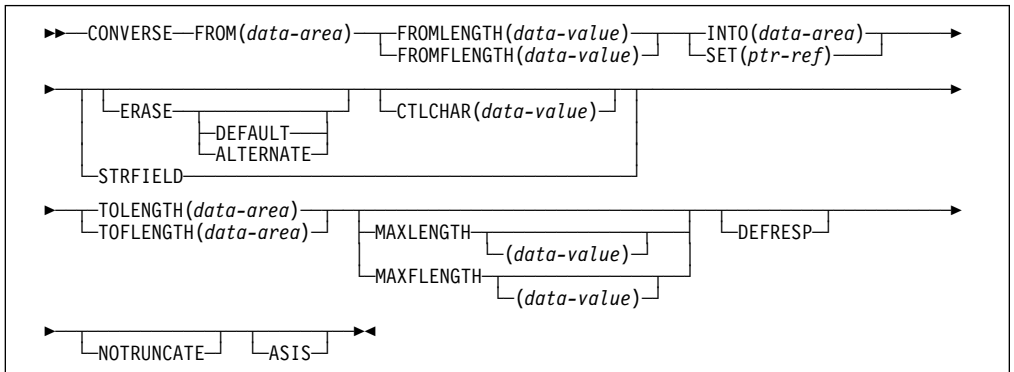
For details of the option PARTNER see “EXEC CICS CONNECT PROCESS” on page 469.

CONVERSE (LUTYPE2/LUTYPE3)

Function

Write data to a 3270-display logical unit (LUTYPE2) or a 3270-printer logical unit (LUTYPE3).

Syntax



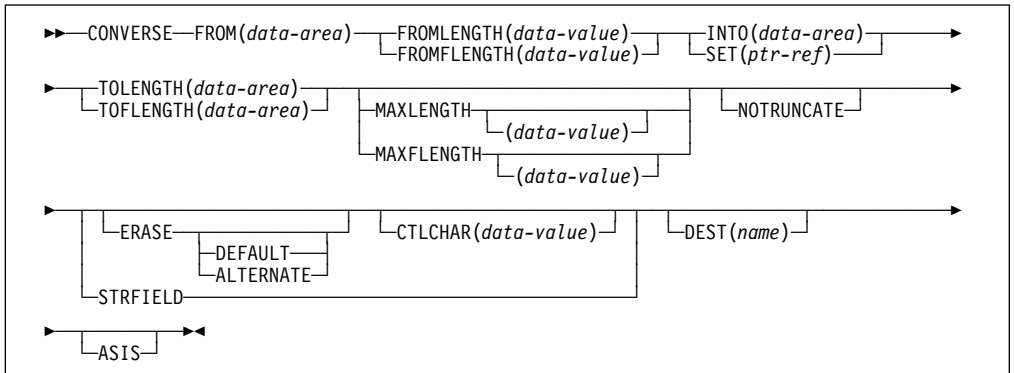
Conditions:
EOC, LENGERR, TERMERR

CONVERSE (3270 display)

Function

Write data to a 3270 information display system (BTAM or TCAM).

Syntax



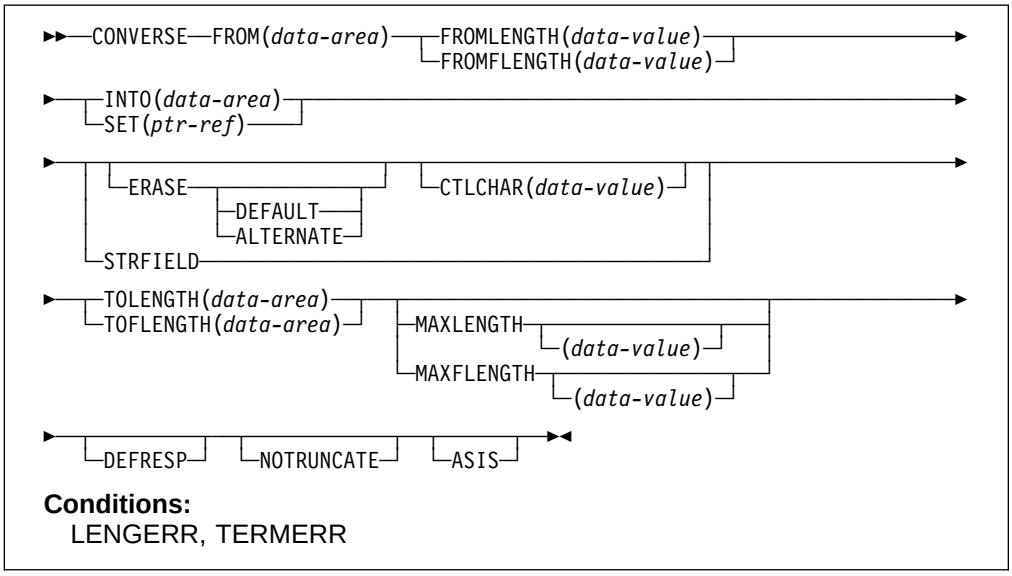
Conditions:
LENGERR

CONVERSE (3270 logical)

Function

Write data to a 3270 logical unit.

Syntax



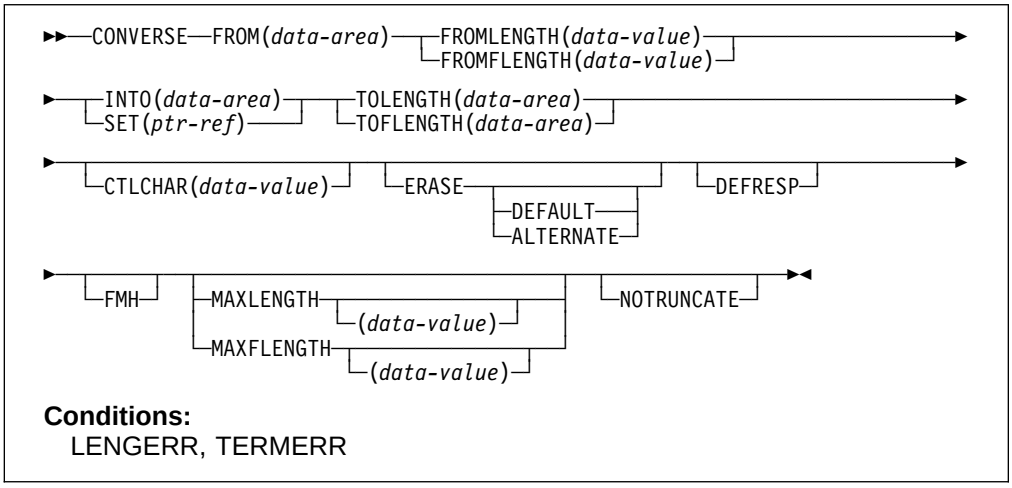
Conditions:
LENGERR, TERMERR

CONVERSE (3650-3270)

Function

Write data to a 3650 logical unit.

Syntax



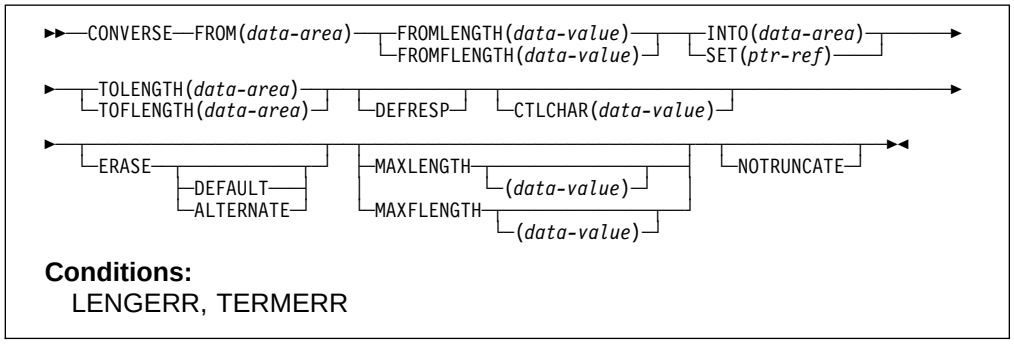
Conditions:
LENGERR, TERMERR

CONVERSE (3790 3270-display)

Function

Write data to a 3790 (3270-display) logical unit.

Syntax



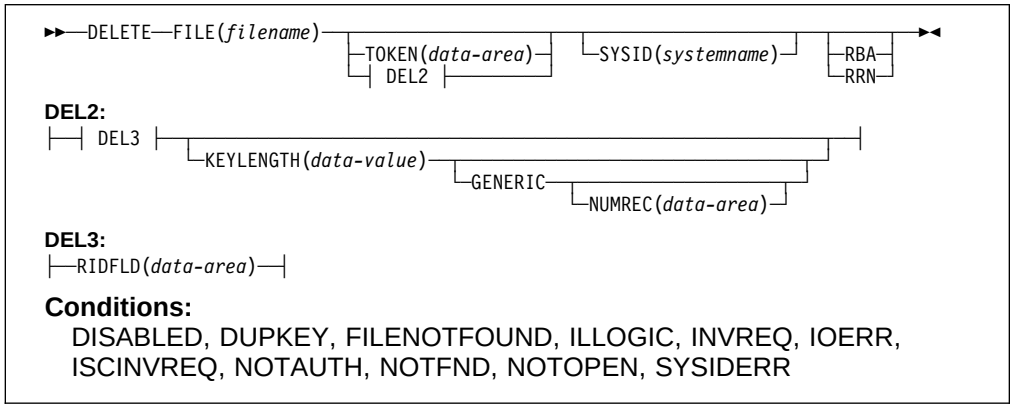
For details of the options ALTERNATE, DEFAULT, and ERASE see “New options on the ERASE parameter” on page 467.

DELETE

Function

Delete a record.

Syntax



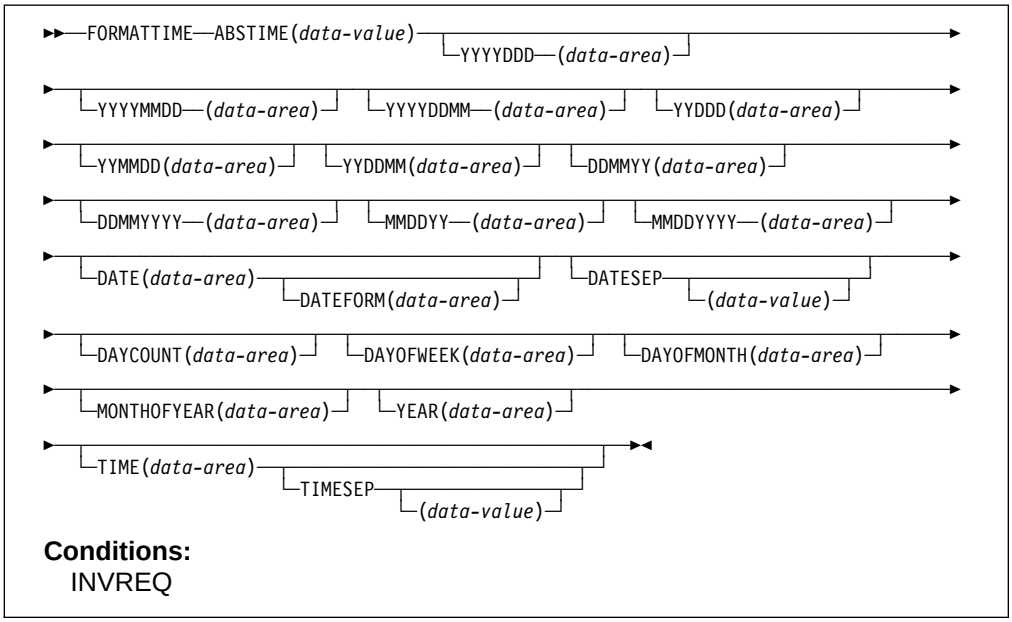
For details of the option TOKEN and the condition INVREQ see “Changes to the application programming interface” on page 465.

FORMATTIME

Function

Transforms the absolute date and time.

Syntax



Conditions:
INVREQ

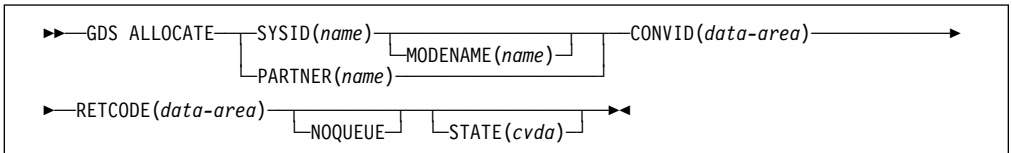
For details of the new keywords see “Four-digit year numbers for dates in the 21st century” on page 463.

GDS ALLOCATE

Function

Acquire a session to a remote APPC logical unit for use by an APPC basic conversation (assembler-language and C/370 programs only).

Syntax



Conditions:
EXEC CICS conditions are never raised on GDS commands. The return code is given in RETCODE (see table following the option descriptions). For a full list of return code values, see the *CICS/ESA Distributed Transaction Programming Guide*.

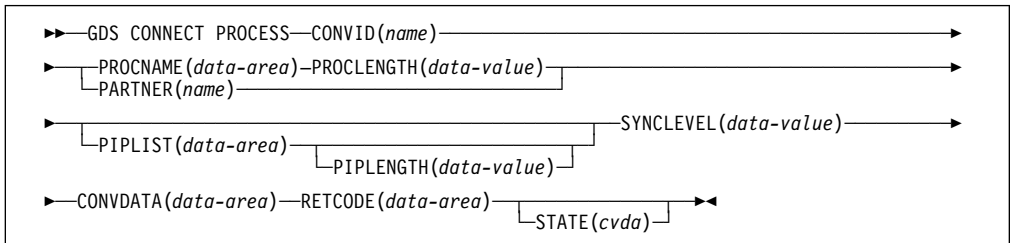
For details of the options MODENAME and PARTNER see “EXEC CICS GDS ALLOCATE” on page 470.

GDS CONNECT PROCESS

Function

Initiate an APPC basic conversation (assembler-language and C/370 programs only).

Syntax



Conditions:

EXEC CICS conditions are never raised on GDS commands. The return code is given in RETCODE (see table following the option descriptions). For a full list of return code values, see the *CICS/ESA Distributed Transaction Programming Guide*.

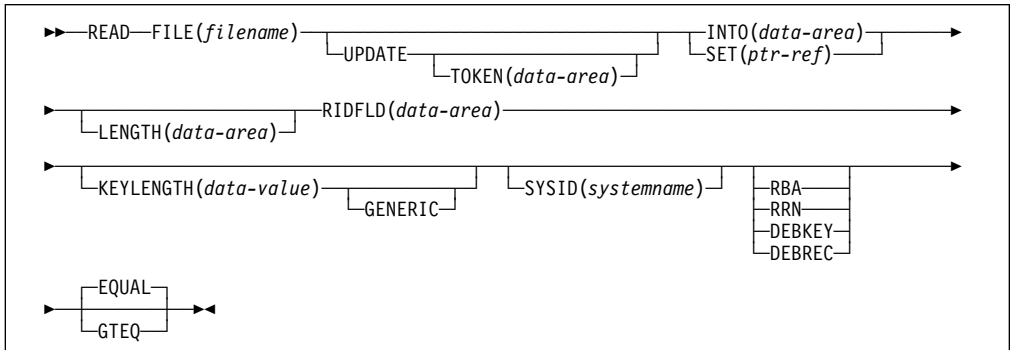
For details of the PARTNER option, see “EXEC CICS GDS CONNECT PROCESS” on page 470.

READ

Function

Read a record from a file.

Syntax



Conditions:

DISABLED, DUPKEY, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, LENGERR, NOTAUTH, NOTFND, NOTOPEN, SYSIDERR

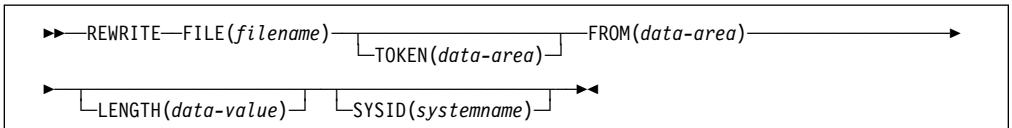
For details of the option TOKEN and the condition INVREQ see “Changes to the application programming interface” on page 465.

REWRITE

Function

Update a record in a file.

Syntax



Conditions:
DUPREC, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, LENGERR, NOSPACE, NOTAUTH, SYSIDERR

For details of the option TOKEN and the condition INVREQ see “Changes to the application programming interface” on page 465.

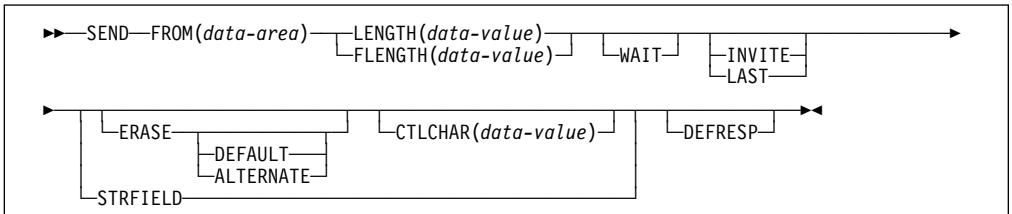
SEND commands

The new options ALTERNATE, DEFAULT, and ERASE are added to the SEND commands for LUTYPE2/LUTYPE3, 3270 display, 3270 logical, 3650-3270, 3790 3270-display and 3270-printer devices. The syntax for all these variants of the SEND command are shown below.

Function

LUTYPE3/LUTYPE3. Write data to a 3270-display logical unit (LUTYPE2) or a 3270-printer logical unit (LUTYPE3).

Syntax

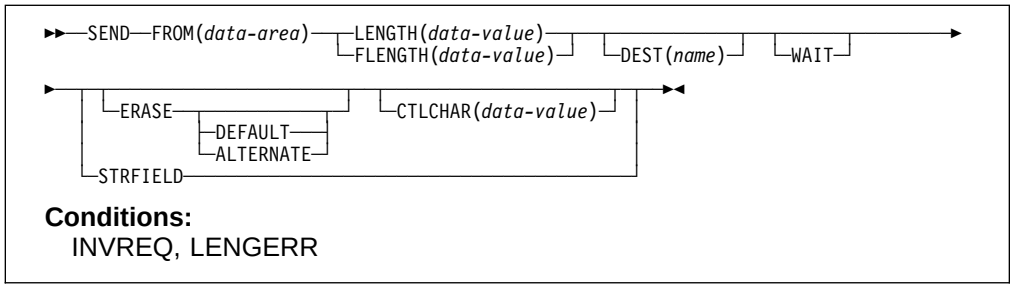


Conditions:
INVREQ, LENGERR, TERMERR

Function

3270 display. Write data to a 3270 information display system (BTAM or TCAM).

Syntax

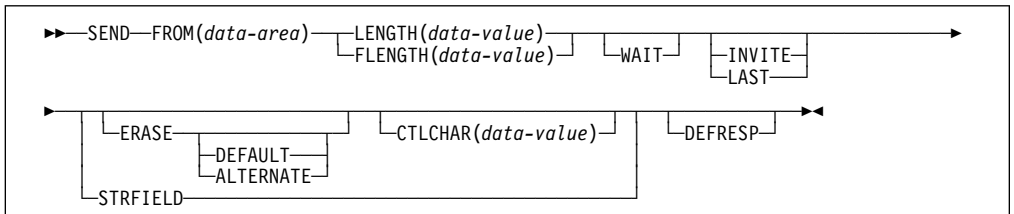


Conditions:
INVREQ, LENGERR

Function

3270 logical. Write data to a 3270 logical unit.

Syntax

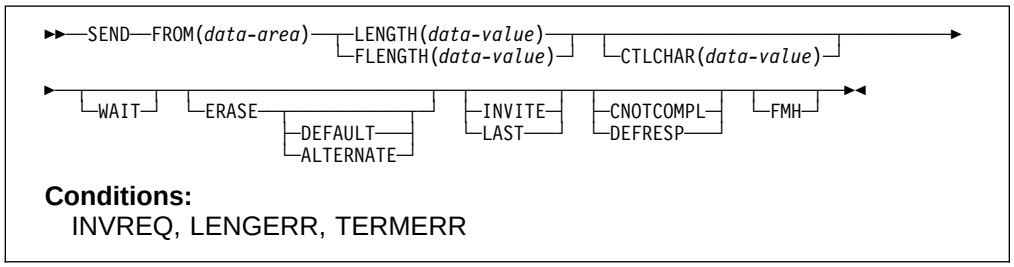


Conditions:
INVREQ, LENGERR, TERMERR

Function

3650-3270. Write data to a 3650 logical unit.

Syntax

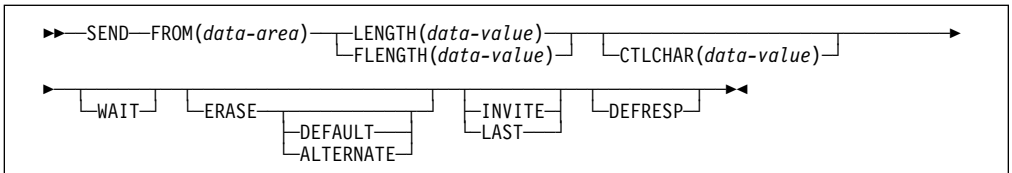


Conditions:
INVREQ, LENGERR, TERMERR

Function

3790 3270-display. Write data to a 3790 logical unit display device.

Syntax

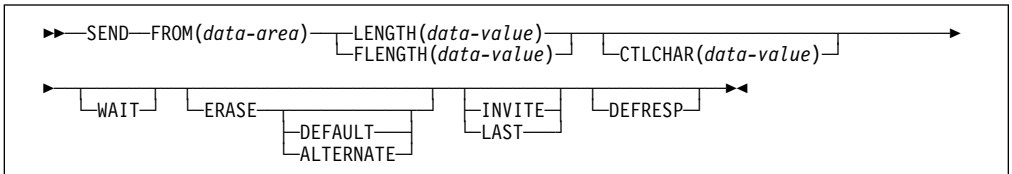


Conditions:
INVREQ, LENGERR, TERMERR

Function

3790 3270-printer Write data to a 3790 logical unit printer device.

Syntax



Conditions:
INVREQ, LENGERR, TERMERR

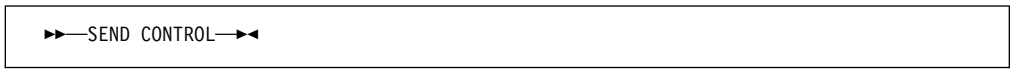
For details of the options ALTERNATE, DEFAULT, and ERASE see “New options on the ERASE parameter” on page 467.

SEND CONTROL

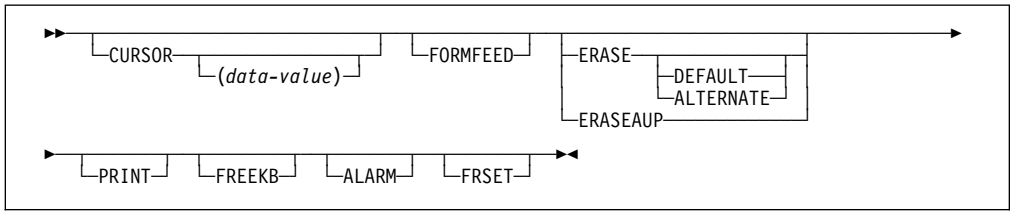
Function

Send device controls to a terminal without map or text data. The keywords are separated into those supported by minimum, standard, and full BMS. For further information about BMS, see the *CICS/ESA Application Programming Guide*.

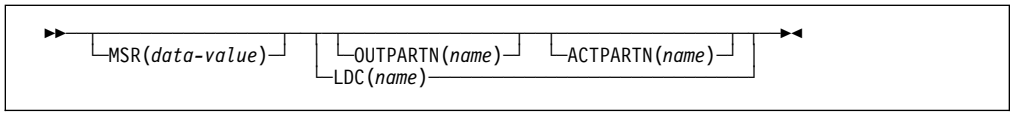
Syntax



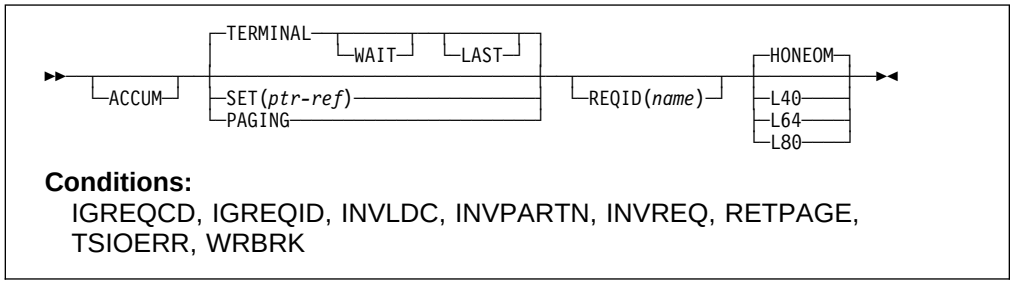
Minimum BMS:



Standard BMS:



Full BMS:



Conditions:
IGREQCD, IGREQID, INVLDC, INVPARTN, INVREQ, RETPAGE, TSIOERR, WRBRK

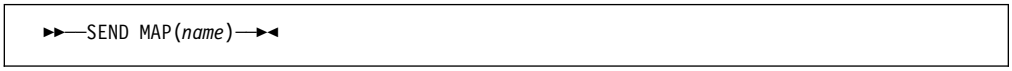
For details of the options ALTERNATE, DEFAULT, and ERASE see “New options on the ERASE parameter” on page 467.

SEND MAP

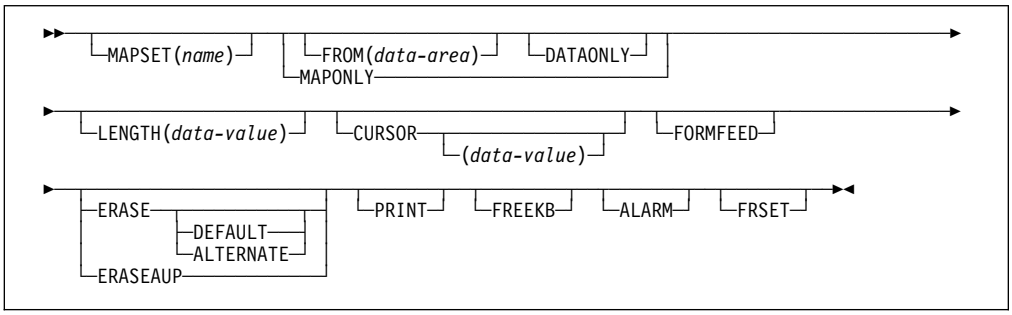
Function

Send mapped output data to a terminal. The keywords are separated into those supported by minimum, standard, and full BMS.

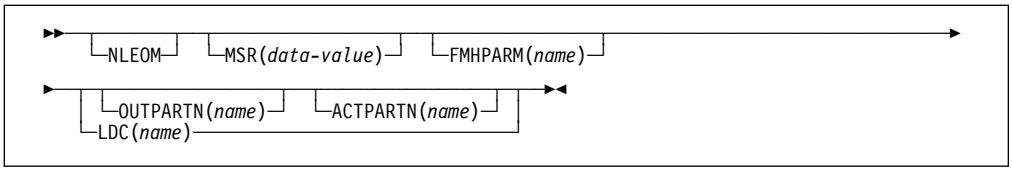
Syntax



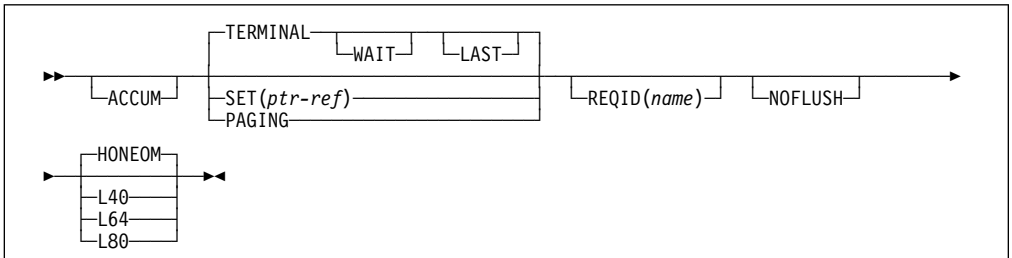
Minimum BMS:



Standard BMS:



Full BMS:



Conditions:
IGREQCD, IGREQID, INVLDC, INVMPSZ, INVPARTN, INVREQ, OVERFLOW, RETPAGE, TSIOERR, WRBRK

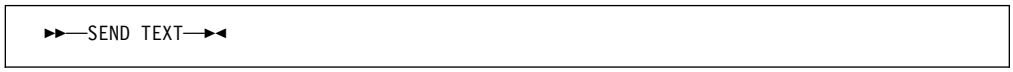
For details of the options ALTERNATE, DEFAULT, and ERASE see “New options on the ERASE parameter” on page 467.

SEND TEXT

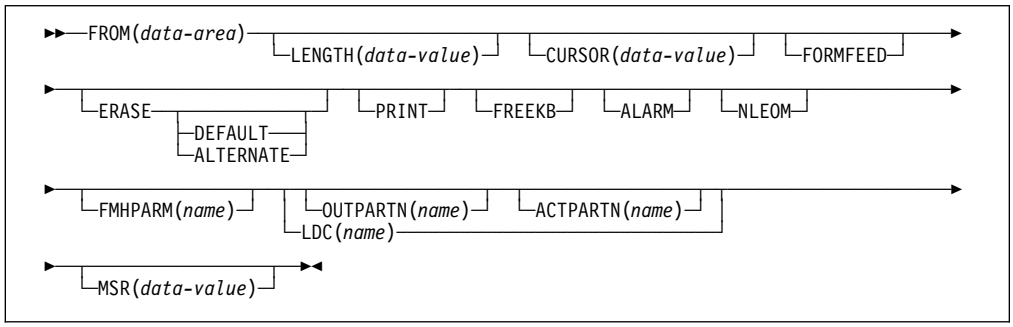
Function

Send data without mapping. The keywords are separated into those supported by standard and full BMS.

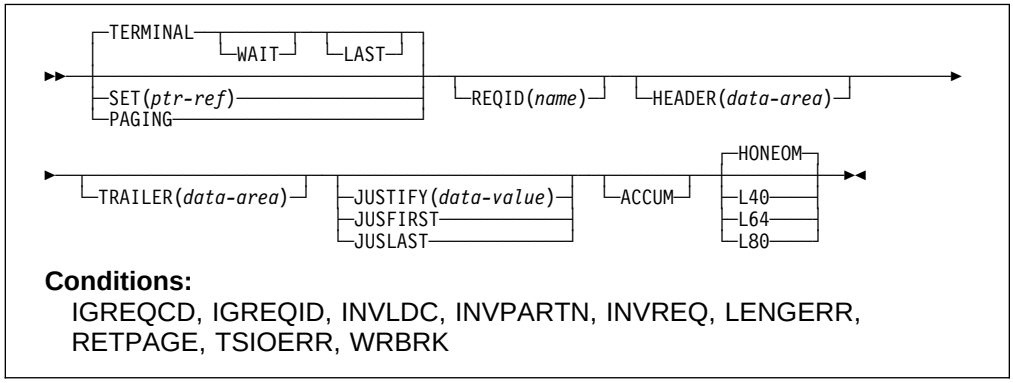
Syntax



Standard BMS:



Full BMS:



Conditions:
IGREQCD, IGREQID, INVLDC, INVPARTN, INVREQ, LENGERR, RETPAGE, TSIOERR, WRBRK

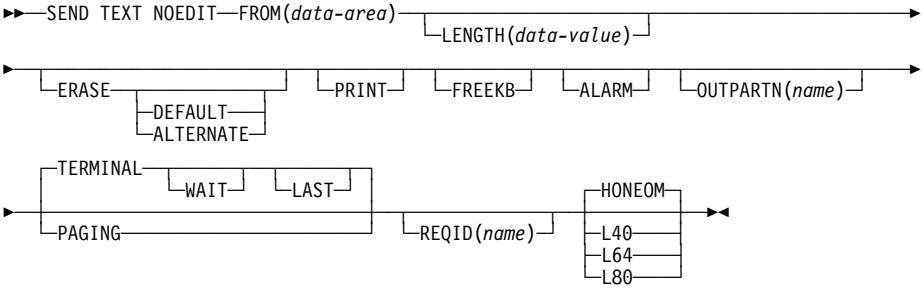
For details of the ALTERNATE, DEFAULT, and ERASE options, see “New options on the ERASE parameter” on page 467.

SEND TEXT NOEDIT

Function

Send a page. Only supplied by full BMS.

Syntax



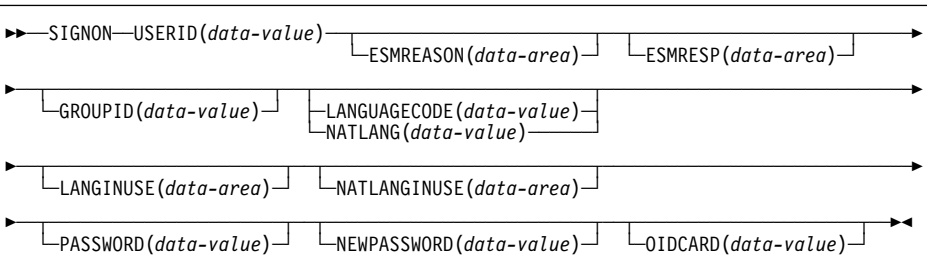
Conditions:
IGREQCD, IGREQID, INVREQ, INVPARTN, TSIOERR, WRBRK

For details of the options ALTERNATE, DEFAULT, and ERASE see “New options on the ERASE parameter” on page 467.

SIGNON

Function

Sign on to a terminal.



Conditions:
INVREQ, NOTAUTH, USERIDERR

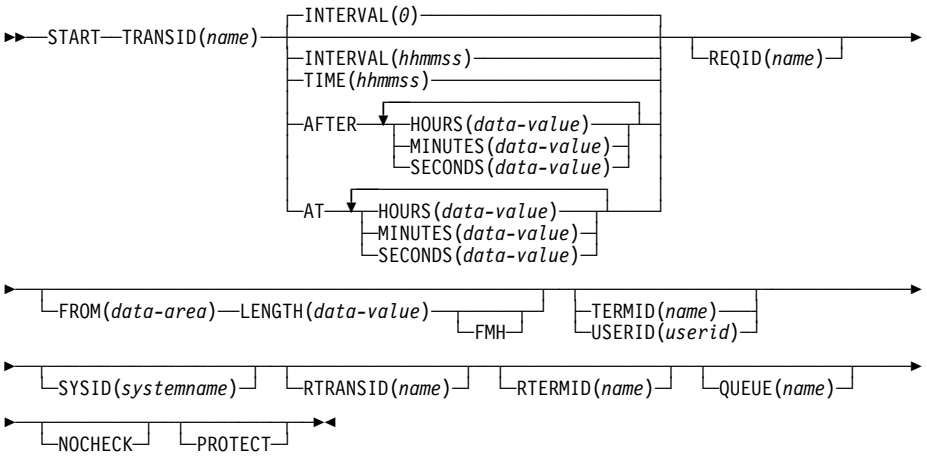
For details of the keywords see “EXEC CICS SIGNON command” on page 410.

START

Function

Start task.

Syntax



Conditions:
INVREQ, IOERR, ISCINVREQ, LENGERR, NOTAUTH, SYSIDERR, TERMIDERR, TRANSIDERR, USERIDERR

For details of the option USERID see “EXEC CICS START command” on page 131.

UNLOCK

Syntax

Conditions:
DISABLED, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, NOTAUTH, NOTOPEN, SYSIDERR

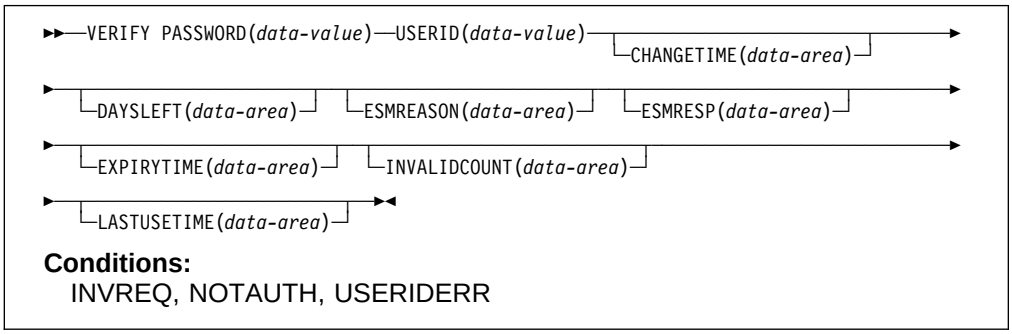
For details of the option TOKEN and the condition INVREQ see “Changes to the application programming interface” on page 465.

VERIFY PASSWORD

Function

Checks a password against the information held for a userid in the RACF database and extracts the relevant data.

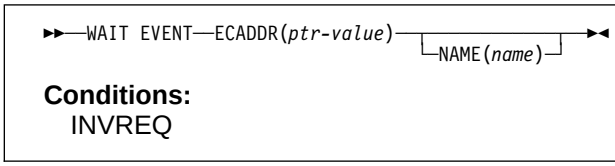
Syntax



For details of the keywords see “EXEC CICS VERIFY PASSWORD command” on page 392.

WAIT EVENT

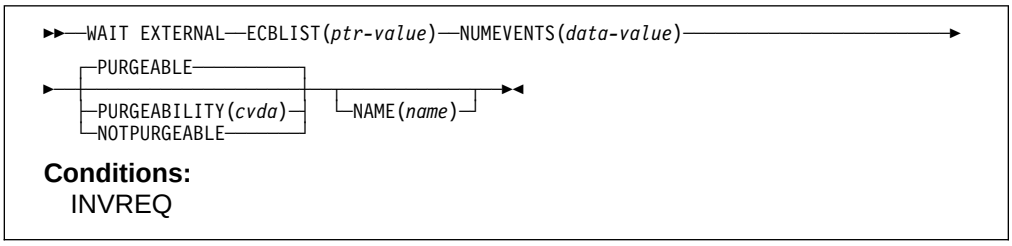
Syntax



For details of the option NAME and the condition INVREQ see “Name option on the wait commands” on page 473.

WAIT EXTERNAL

Syntax

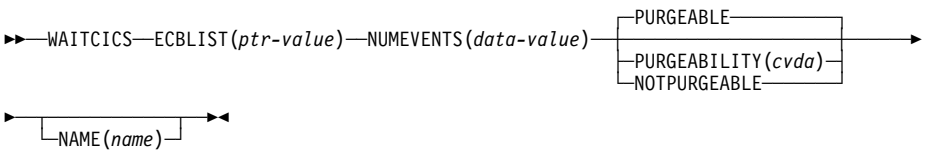


For details of the option NAME and the condition INVREQ see “Name option on the wait commands” on page 473.

WAITCICS

Function

Syntax



Conditions:
INVREQ

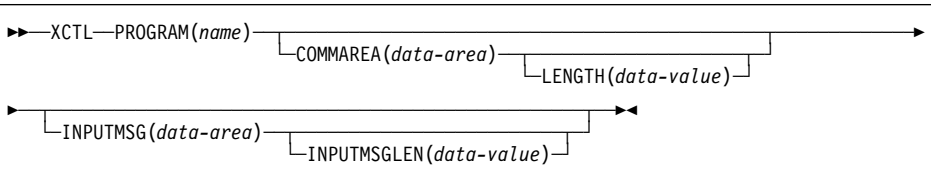
For details of the option NAME and the condition INVREQ see “Name option on the wait commands” on page 473.

XCTL

Function

Transfer program control.

Syntax



Conditions:
INVREQ, LENGERR, NOTAUTH, PGMIDERR

For details of the condition LENGERR see “Changes to COMMAREA processing” on page 334.

Chapter 35. Changes to the system programming interface (SPI)

This chapter shows all the changes to the system programming interface made in CICS/ESA 4.1. The syntax diagrams show all the parameters, with the changed or new parameters marked with a vertical line to the left.

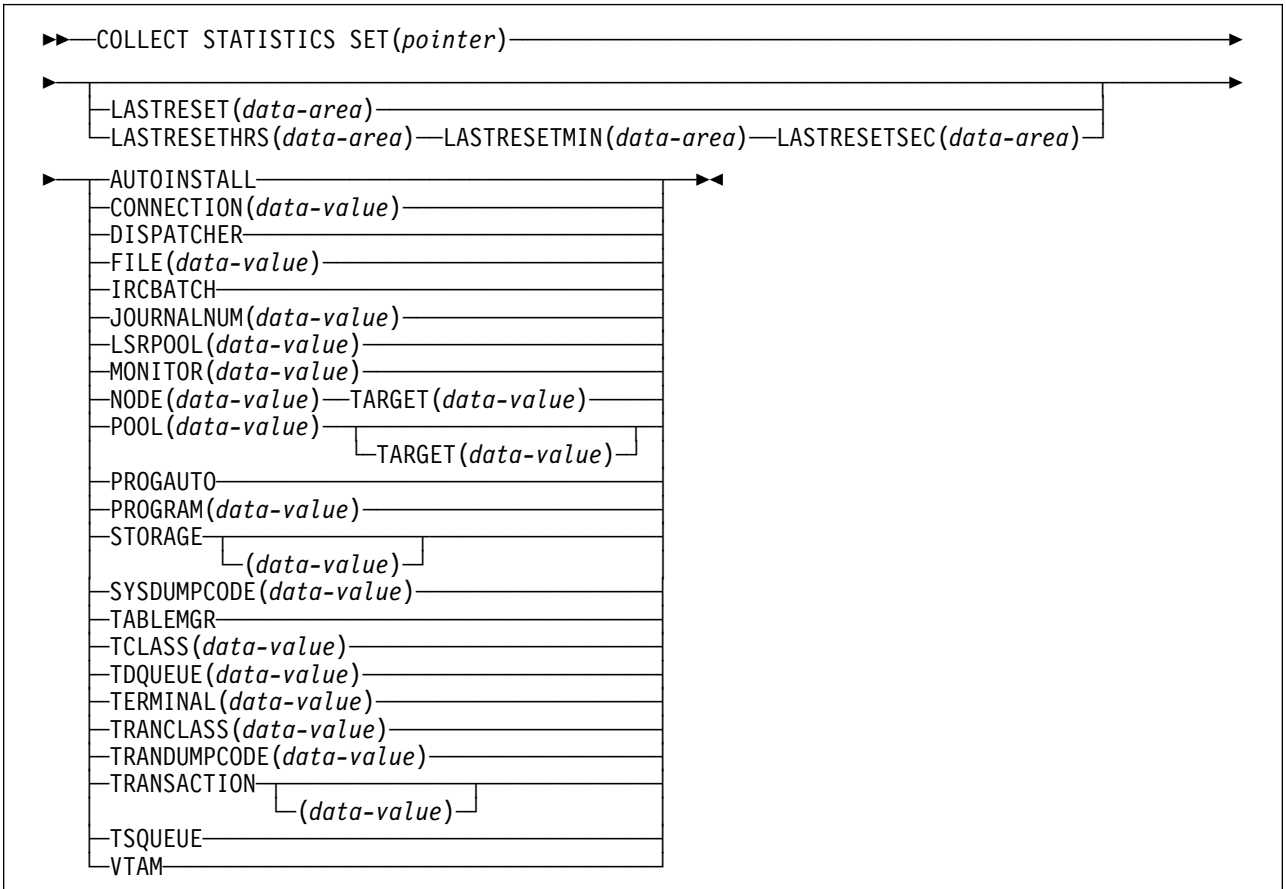
This chapter contains General-use Programming Interface information.

COLLECT STATISTICS

Function

Returns to the invoking application the current statistics for a single named resource, or the global statistics for a named resource type.

Syntax



Conditions:
INVREQ, IOERR, NOTAUTH, NOTFND

For details of the keywords PROGAUTO, STORAGE, TRANSACTION, TRANCLASS, POOL, and NODE TARGET, see “EXEC CICS COLLECT STATISTICS” on page 301.

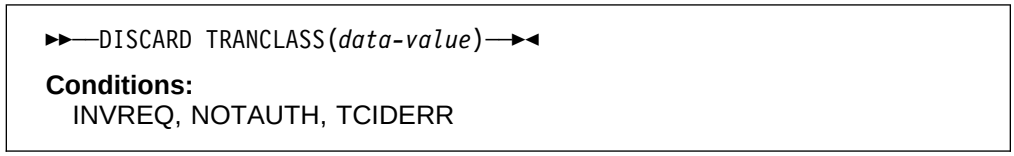
DISCARD TRANCLASS

Function

Removes the installed name of the transaction class from storage and the CICS catalog.

Syntax

For details of the keyword TRANCLASS, see “EXEC CICS DISCARD TRANCLASS command” on page 360.

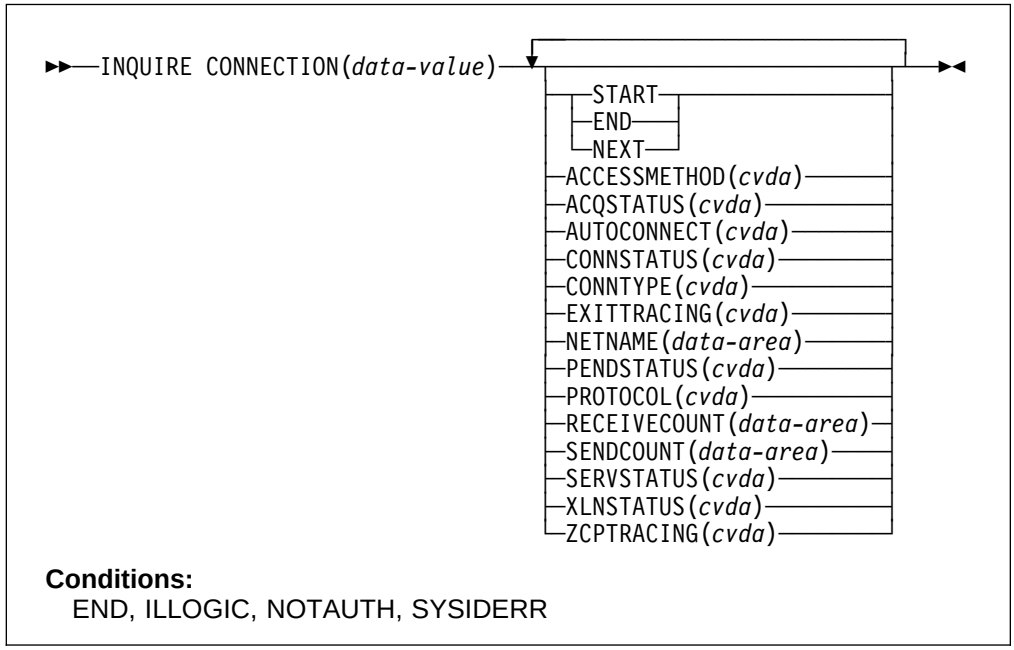


INQUIRE CONNECTION

Function

Retrieves information about a named connection to a remote system. The remote system can be another CICS region.

Syntax



For details of the keywords CONNTYPE, PROTOCOL, RECEIVECOUNT, and SENDCOUNT, see “EXEC CICS INQUIRE CONNECTION command” on page 171. For details of the keyword CONNSTATUS, see “EXEC CICS INQUIRE CONNECTION command” on page 218.

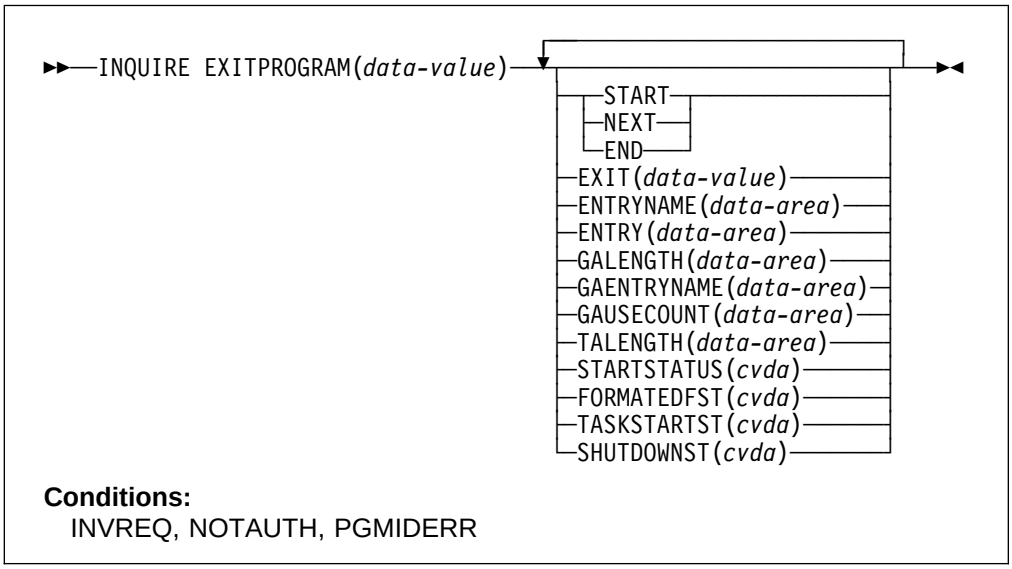
INQUIRE EXITPROGRAM

Function

Returns information about the global user exit programs at exit points and task-related user exit programs.

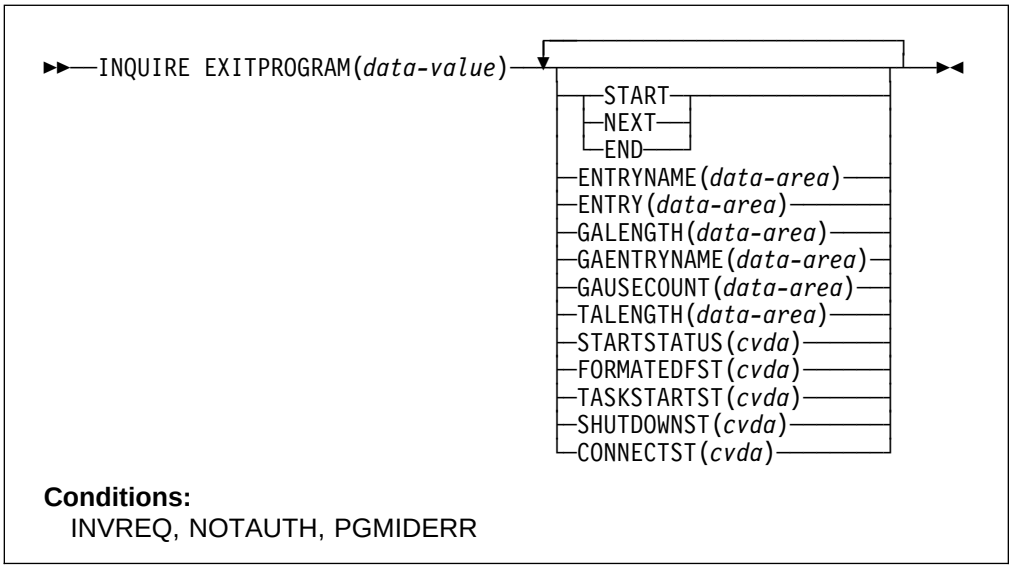
Syntax

For global user exit programs:



Conditions:
INVREQ, NOTAUTH, PGMIDERR

For task-related user exit programs:



Conditions:
INVREQ, NOTAUTH, PGMIDERR

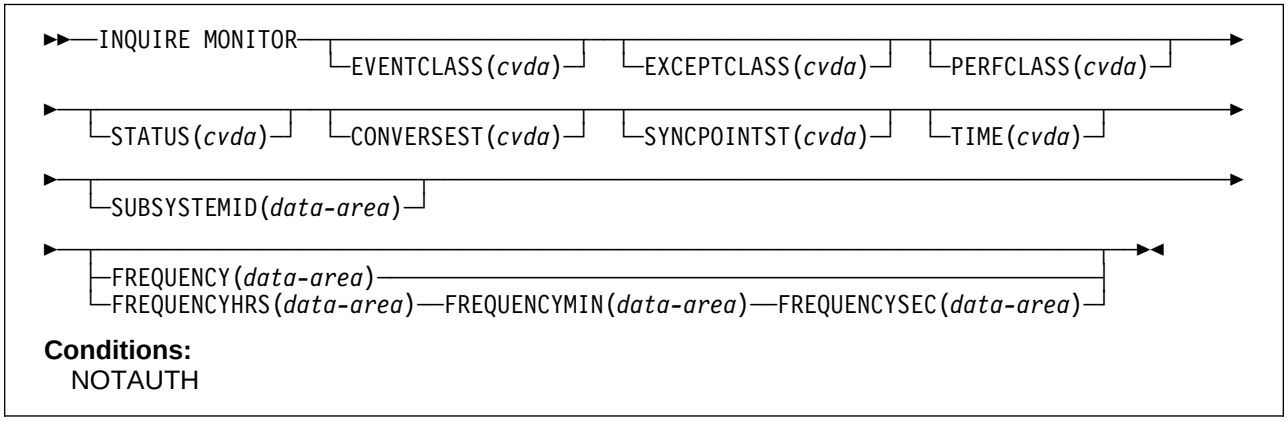
For details of the keywords and conditions for this command, see “EXEC CICS INQUIRE EXITPROGRAM command” on page 219.

INQUIRE MONITOR

Function

The INQUIRE MONITOR command allows you to find out whether CICS is accumulating monitoring data for executing transactions, and to discover which monitoring data classes are active.

Syntax



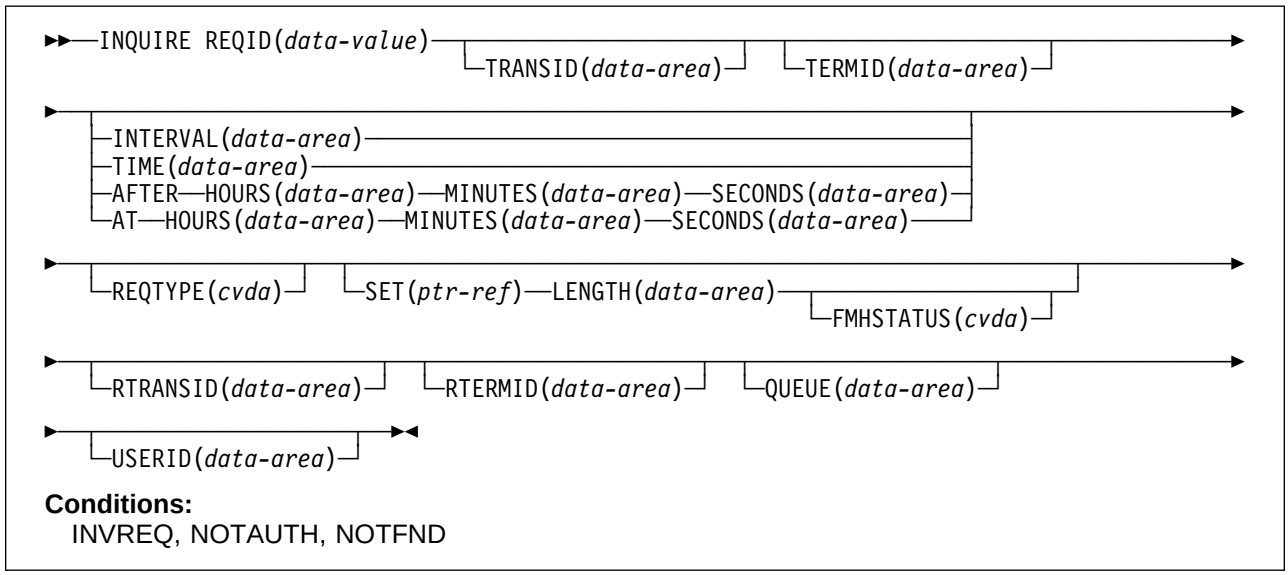
For details of the keywords CONVERSEST, FREQUENCY, FREQUENCYHRS, FREQUENCYMIN, FREQUENCYSEC, SUBSYSTEMID, SYNCPOINTST, and TIME see “EXEC CICS INQUIRE MONITOR command” on page 305.

INQUIRE REQID

Function

Recover data.

Syntax

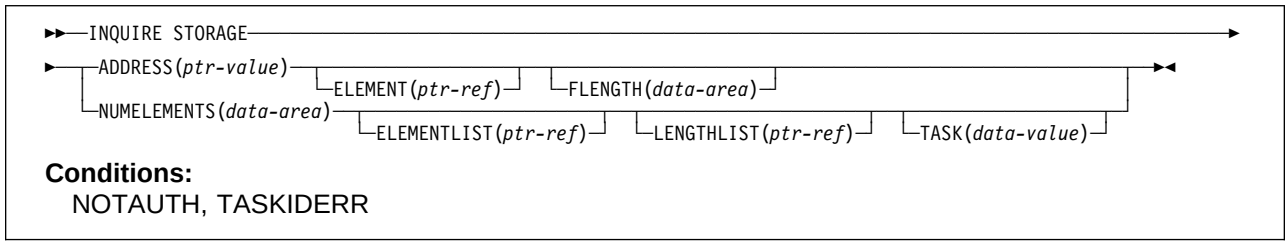


For details of the keywords and conditions for this command, see "EXEC CICS INQUIRE REQID command" on page 223.

INQUIRE STORAGE

Function: Obtain from CICS information about elements of task-lifetime storage.

Syntax



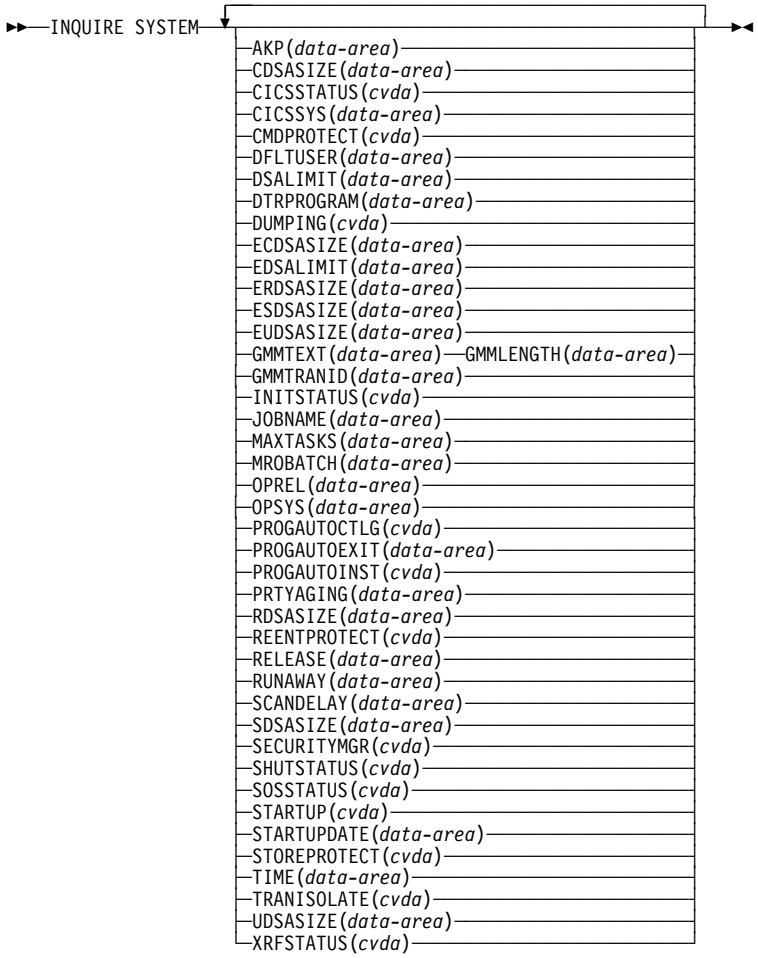
For details of the keywords see "EXEC CICS INQUIRE STORAGE command" on page 32.

INQUIRE SYSTEM

Function

Returns information about the local CICS system.

Syntax



Conditions:
ADDRESSERR, NOTAUTH

For details of the keywords CMDPROTECT, DSALIMIT, EDSALIMIT, EDSASIZE, RDSASIZE, REENTPROTECT, SDSASIZE, SOSSTATUS, and TRANISOLATE see “EXEC CICS INQUIRE SYSTEM command” on page 34.

For details of the keywords PROGAUTOCTL, PROGAUTOEXIT, and PROGAUTOINST see “EXEC CICS INQUIRE SYSTEM command” on page 255.

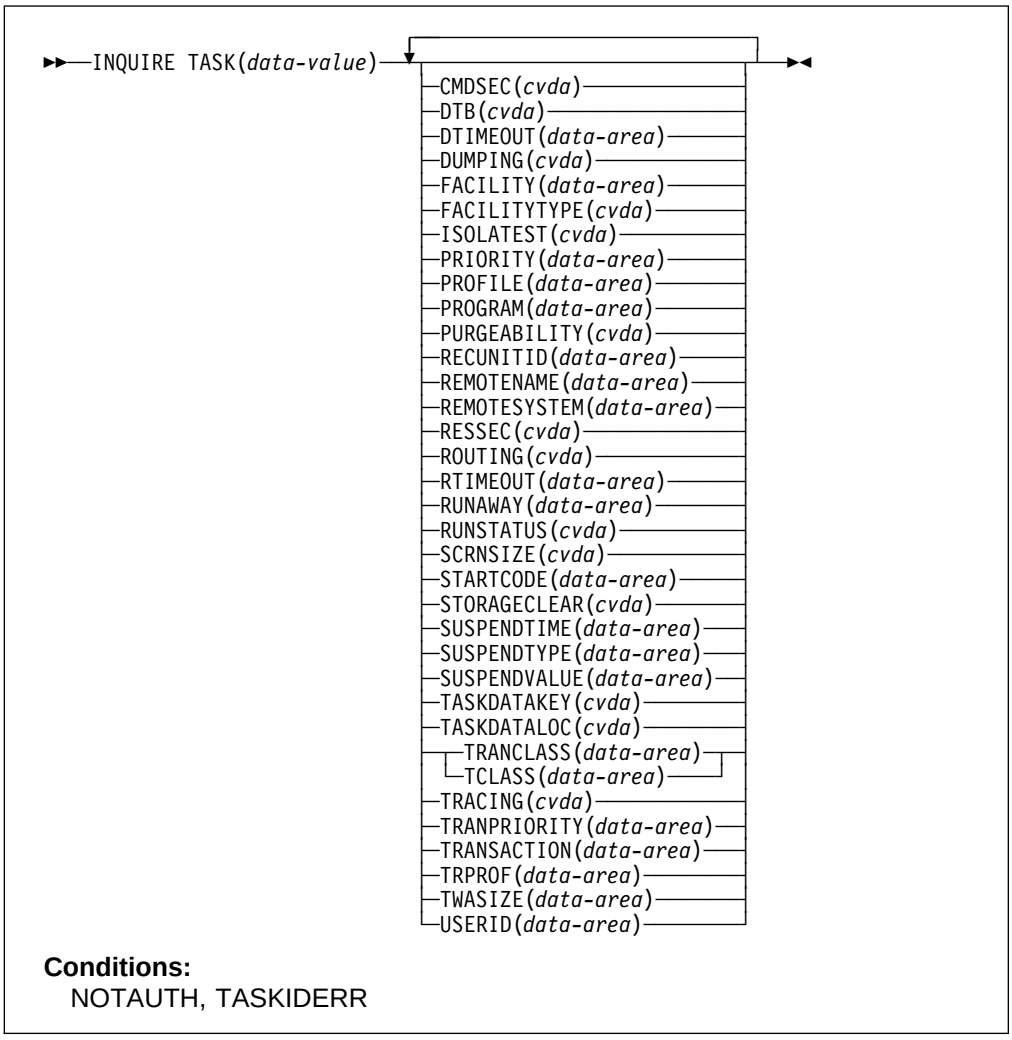
For details of the keywords GMMTEXT, GMMLENGTH, INITSTATUS, SHUTSTATUS, and STARTUPDATE, see “EXEC CICS INQUIRE SYSTEM command” on page 228.

INQUIRE TASK

Function

Returns information about user tasks. User tasks are tasks associated with user-defined transactions and tasks associated with the CICS-supplied transactions that are normally invoked by an operator.

Syntax



Conditions:
NOTAUTH, TASKIDERR

INQUIRE TASK options

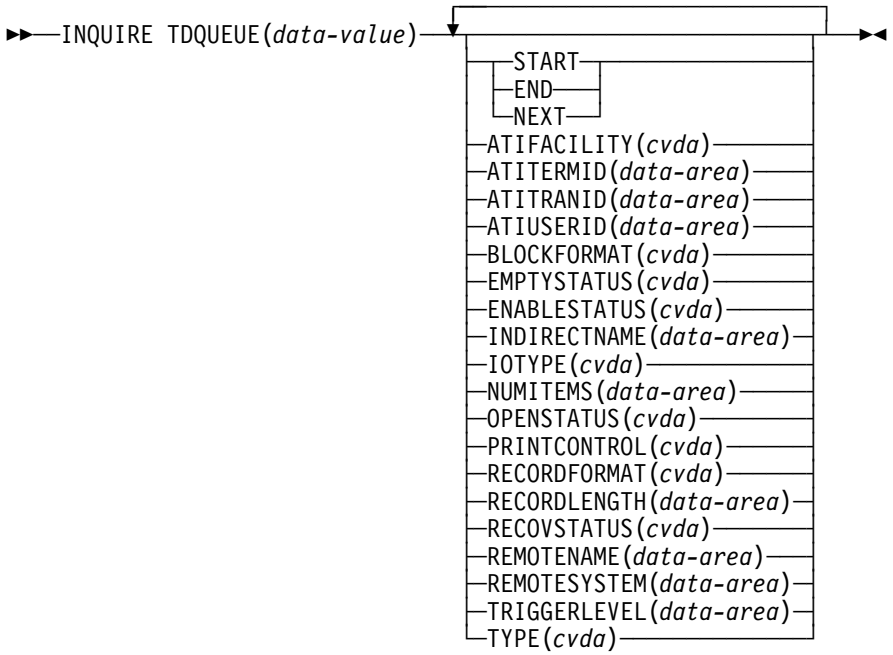
For details of the changed keywords see “EXEC CICS INQUIRE TASK command” on page 348.

INQUIRE TDQUEUE

Function

Returns information about a named transient data queue.

Syntax



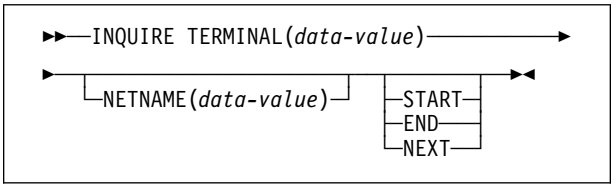
Conditions:
END, ILLOGIC, NOTAUTH, QIDERR

For details of the keyword ATIUSERID see “EXEC CICS INQUIRE TDQUEUE command” on page 132.

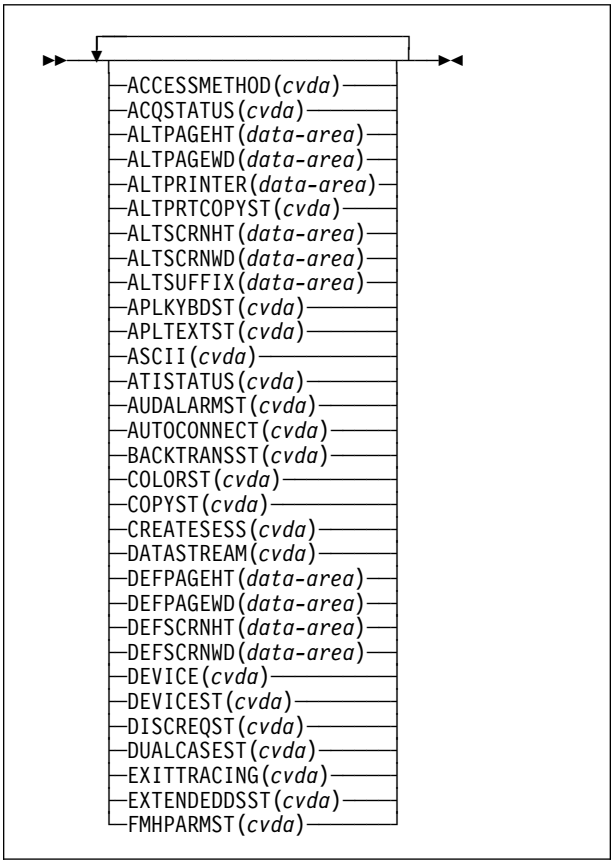
INQUIRE TERMINAL/NETNAME

Function: Returns information about a named terminal.

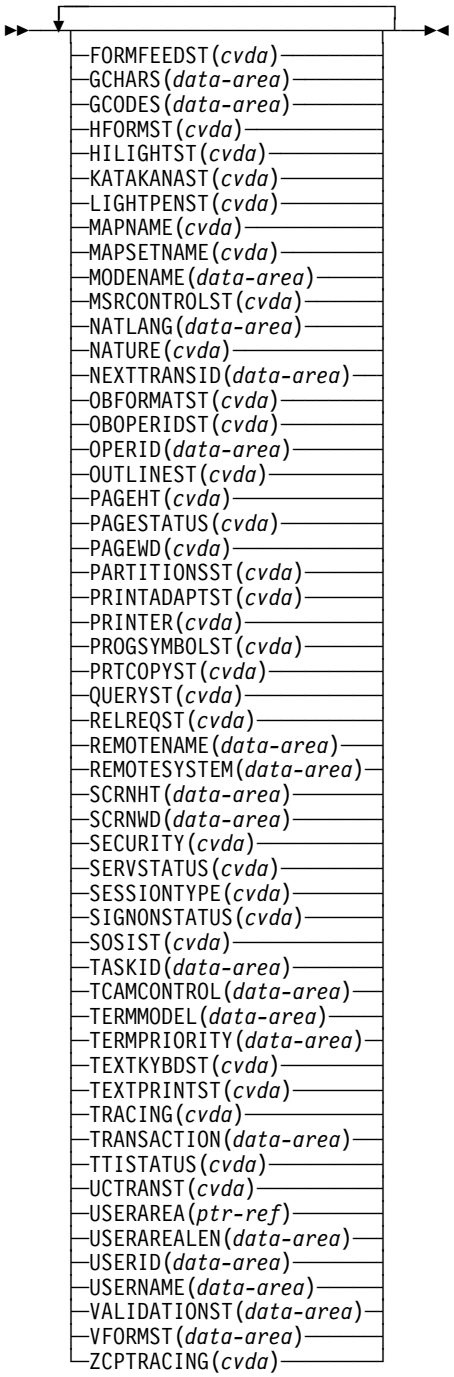
Syntax



The following options apply to both the INQUIRE TERMINAL and the INQUIRE NETNAME command.



+
+



Conditions:
END, ILLOGIC, NOTAUTH, TERMIDERR

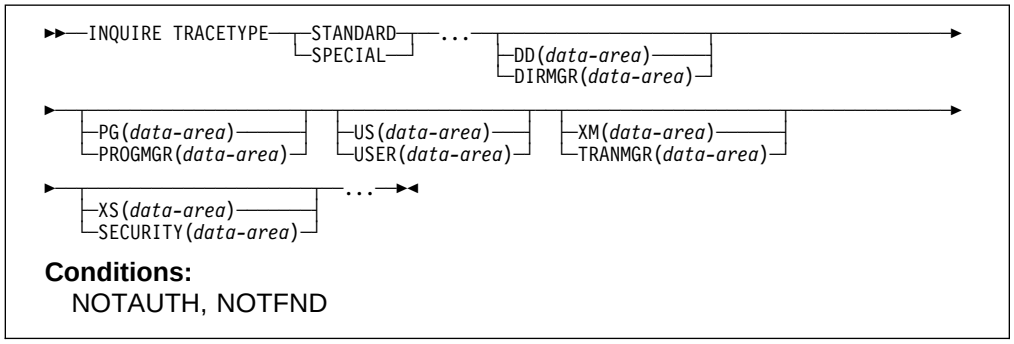
For details of the keywords ASCII, AUTOCONNECT, DATASTREAM, and, DEVICEST see “EXEC CICS INQUIRE TERMINAL | NETNAME command” on page 229. For details of the keywords MAPNAME and MAPSETNAME, see “MAPNAME and MAPSETNAME options on INQUIRE and SET TERMINAL” on page 473.

INQUIRE TRACETYPE

Function

Retrieves information on trace levels for a named CICS component.

Syntax



Conditions:
NOTAUTH, NOTFND

For details of the keywords DD and DIRMGR see “INQUIRE TRACETYPE command” on page 328.

For details of the keywords PG and PROGMGR see “EXEC CICS INQUIRE TRACETYPE command” on page 335.

For details of the keywords US and USER see “Changes to the system programming interface” on page 405.

For details of the keywords XM and TRANMGR see “EXEC CICS INQUIRE and SET TRACETYPE command” on page 353.

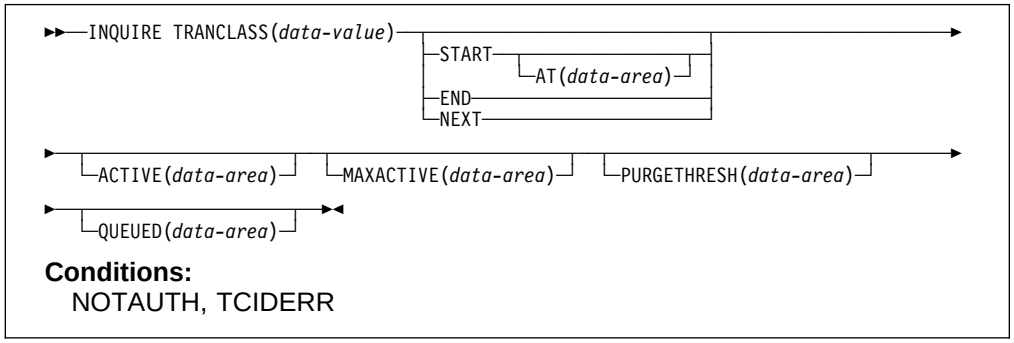
For details of the keywords XS and SECURITY see “Changes to the system programming interface” on page 395.

INQUIRE TRANCLASS

Function

Reports on the status of transaction classes.

Syntax



Conditions:
NOTAUTH, TCIDERR

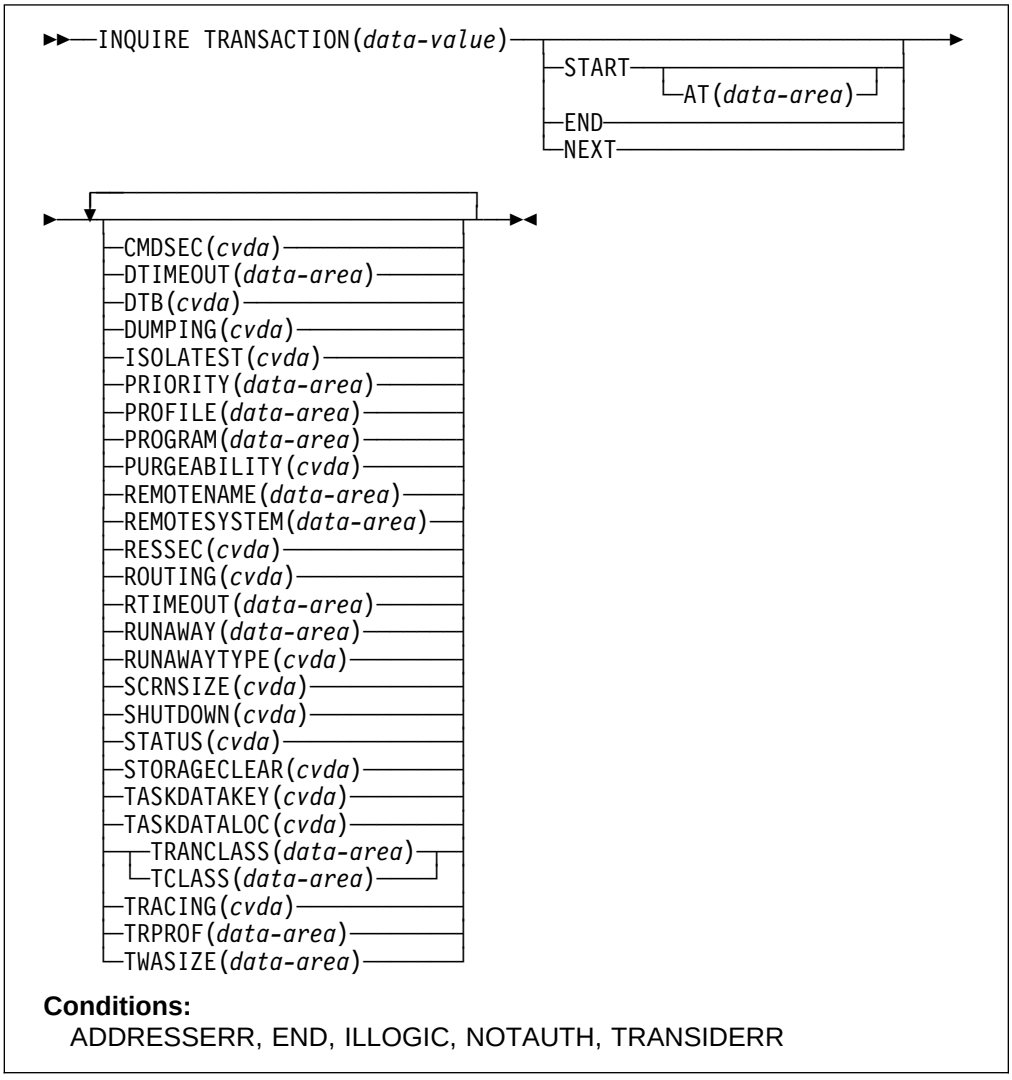
For details of the keywords ACTIVE, MAXACTIVE, PURGETHRESH, and QUEUED see “EXEC CICS INQUIRE TRANCLASS command” on page 357.

INQUIRE TRANSACTION

Function

Returns information about a named transaction.

Syntax



Conditions:
ADDRESSERR, END, ILLOGIC, NOTAUTH, TRANSIDERR

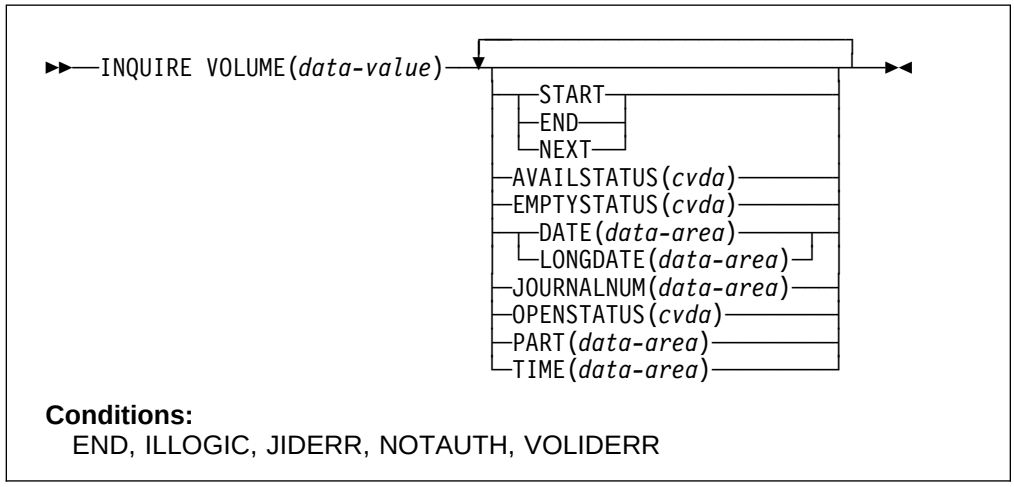
For details of the keyword ISOLATEST see “INQUIRE TASK and INQUIRE TRANSACTION options” on page 38.

For details of the keywords RUNAWAY, RUNAWAYTYPE, SHUTDOWN, STORAGECLEAR, TCLASS, and TRANSCLASS see “EXEC CICS INQUIRE TRANSACTION command” on page 354.

INQUIRE VOLUME

Function
Retrieves information about a named volume.

Syntax



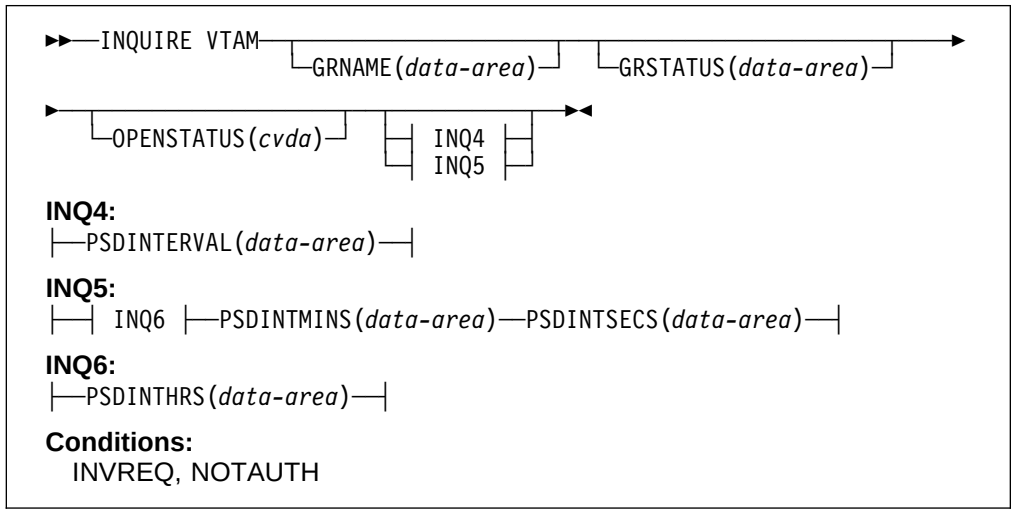
Conditions:
END, ILLOGIC, JIDERR, NOTAUTH, VOLIDERR

For details of the LONGDATE keyword, see "INQUIRE VOLUME LONGDATE(data-area)" on page 464.

INQUIRE VTAM

Function
Inquire on the state of the connection between CICS and VTAM and also on the persistent session delay interval (PSDI).

Syntax



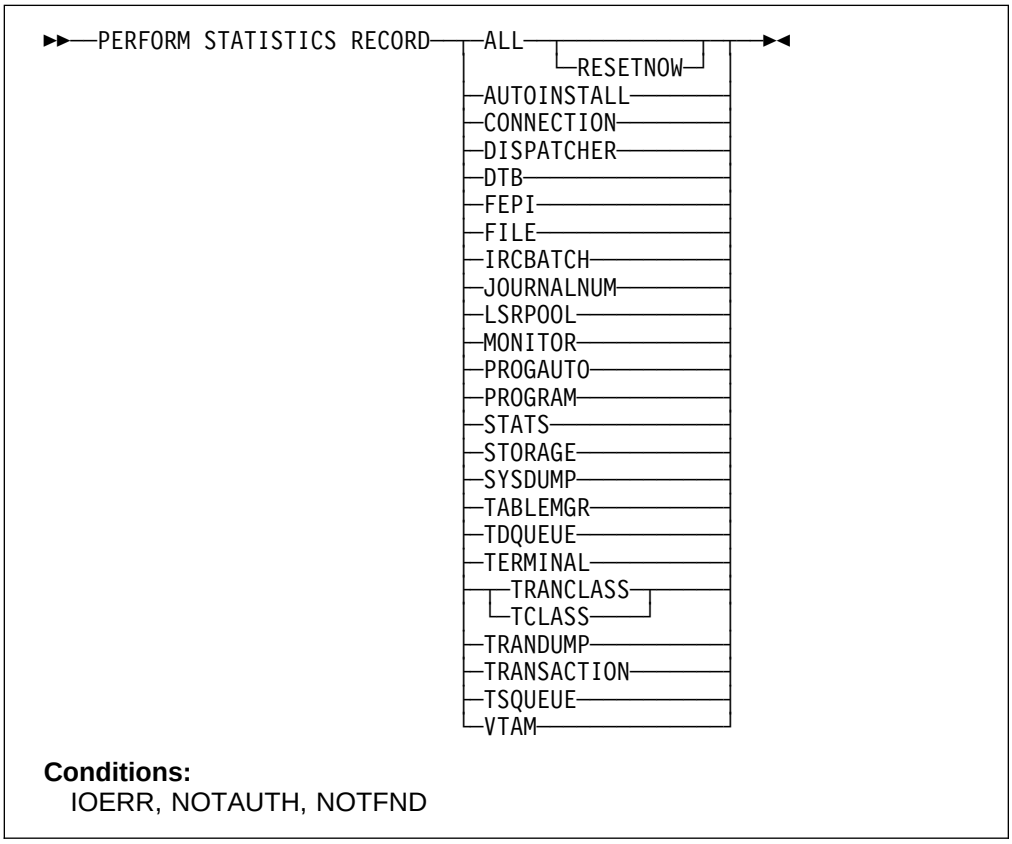
Conditions:
INVREQ, NOTAUTH

For details of the GRNAME and GRSTATUS keywords, see "Changes to CICS-supplied transactions" on page 491.

For details of the keywords PSDINTERVAL, PSDINTHRS, PSDINTMINS, PSDINTSECS see “EXEC CICS INQUIRE VTAM command” on page 62.

PERFORM STATISTICS RECORD

Syntax



Conditions:
IOERR, NOTAUTH, NOTFND

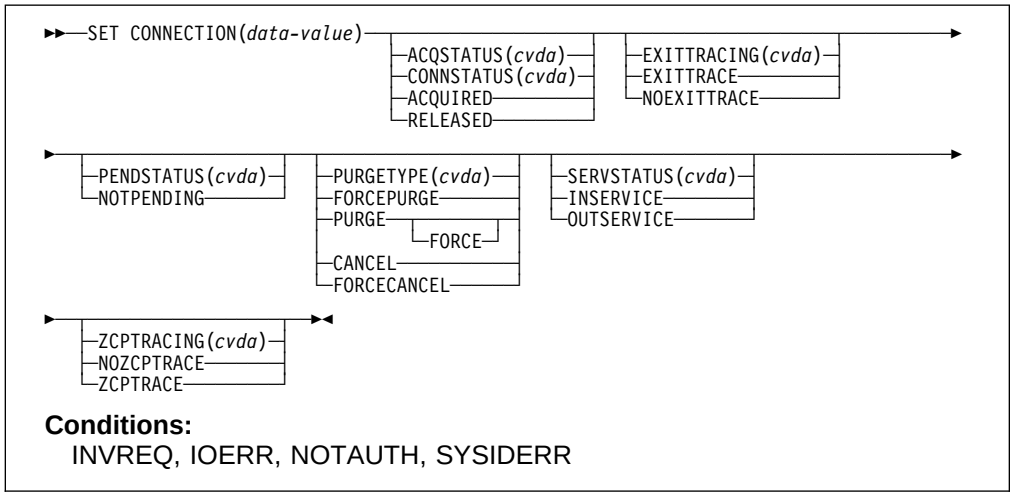
For details of the keywords FEPI, PROGAUTO, and TRANCLASS see “EXEC CICS PERFORM STATISTICS RECORD” on page 304.

SET CONNECTION

Function

To change some connection attributes, and cancel outstanding AIDs.

Syntax



Conditions:
INVREQ, IOERR, NOTAUTH, SYSIDERR

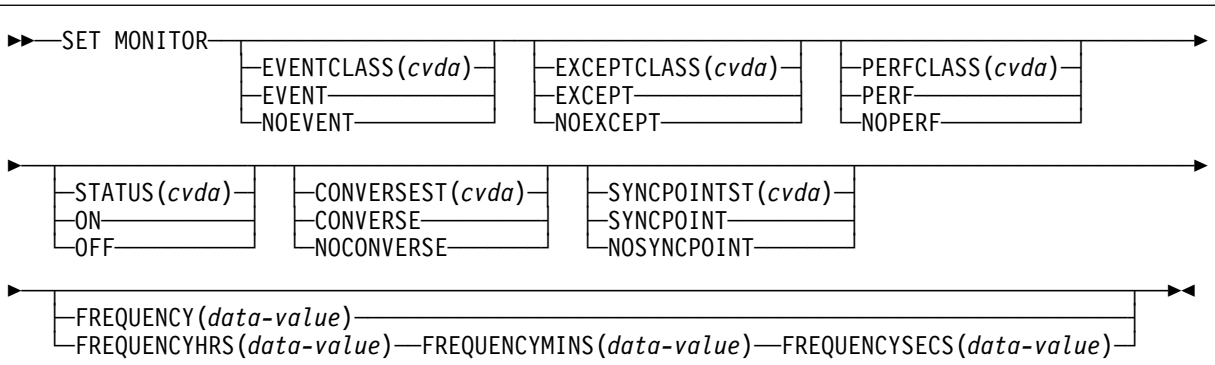
For details of the keywords CANCEL and FORCECANCEL see “EXEC CICS SET CONNECTION command” on page 241.

SET MONITOR

Function

Switch CICS monitoring on or off and select the classes of monitoring data to be collected.

Syntax



Conditions:
INVREQ, NOTAUTH

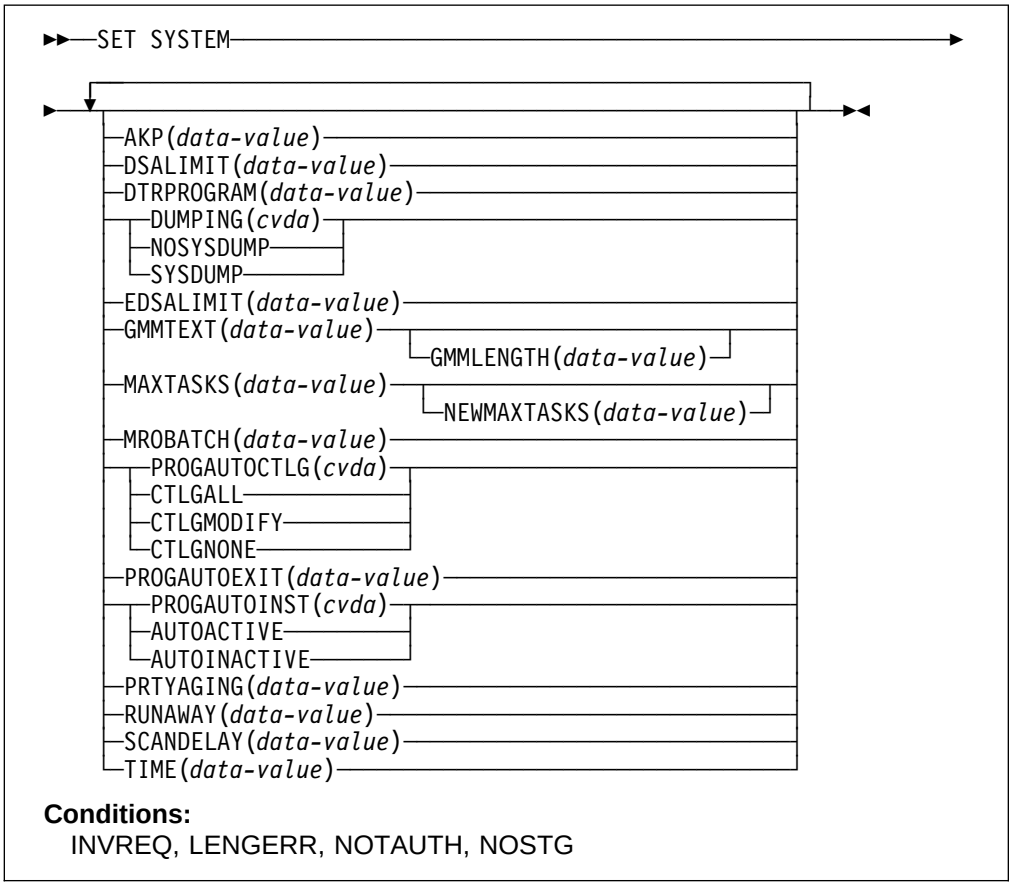
For details of the keywords CONVERSEST, FREQUENCY, FREQUENCYHRS, FREQUENCYMIN, FREQUENCYSEC, SYNCPOINTST, and TIME see “EXEC CICS SET MONITOR” on page 307.

SET SYSTEM

Function

Change the value of some of the system attributes.

Syntax



Conditions:
INVREQ, LENGERR, NOTAUTH, NOSTG

For details of the keywords DSALIMIT and EDSALIMIT see “EXEC CICS SET SYSTEM command” on page 36.

For details of the keywords GMMLENGTH, and GMMTEXT see “EXEC CICS SET SYSTEM command” on page 230.

For details of the keywords PROGAUTOCTL, PROGAUTOEXIT, and PROGAUTOINST see “EXEC CICS SET SYSTEM command” on page 256.

SET TASK

No change is made to the syntax of the EXEC CICS SET TASK command, however the action taken and response returned as a result of an EXEC CICS SET TASK PURGED|FORCEPURGED may be different between CICS/ESA Version 3 and CICS/ESA Version 4.

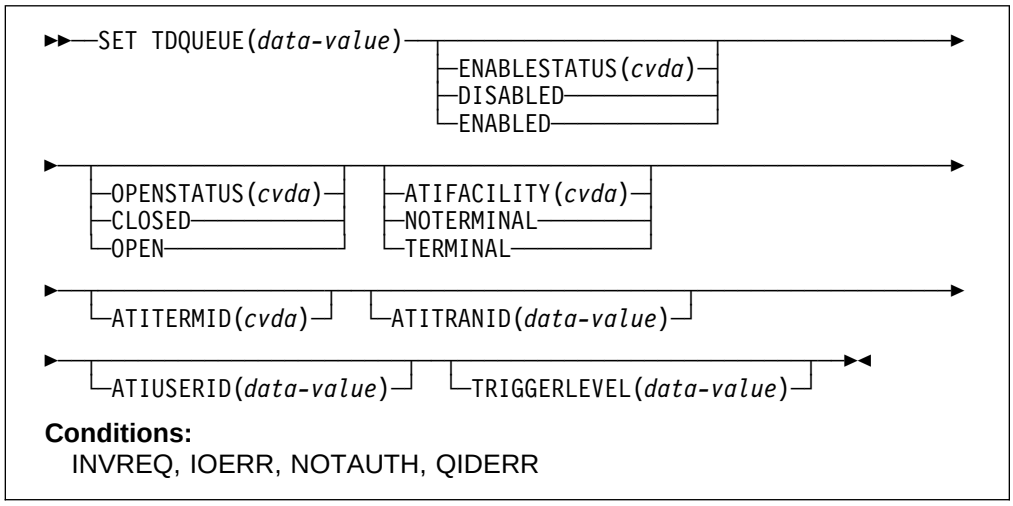
For details of the keyword PURGETYPE see “EXEC CICS SET TASK command” on page 352.

SET TDQUEUE

Function

Change the value of some of the system attributes of a transient data queue.

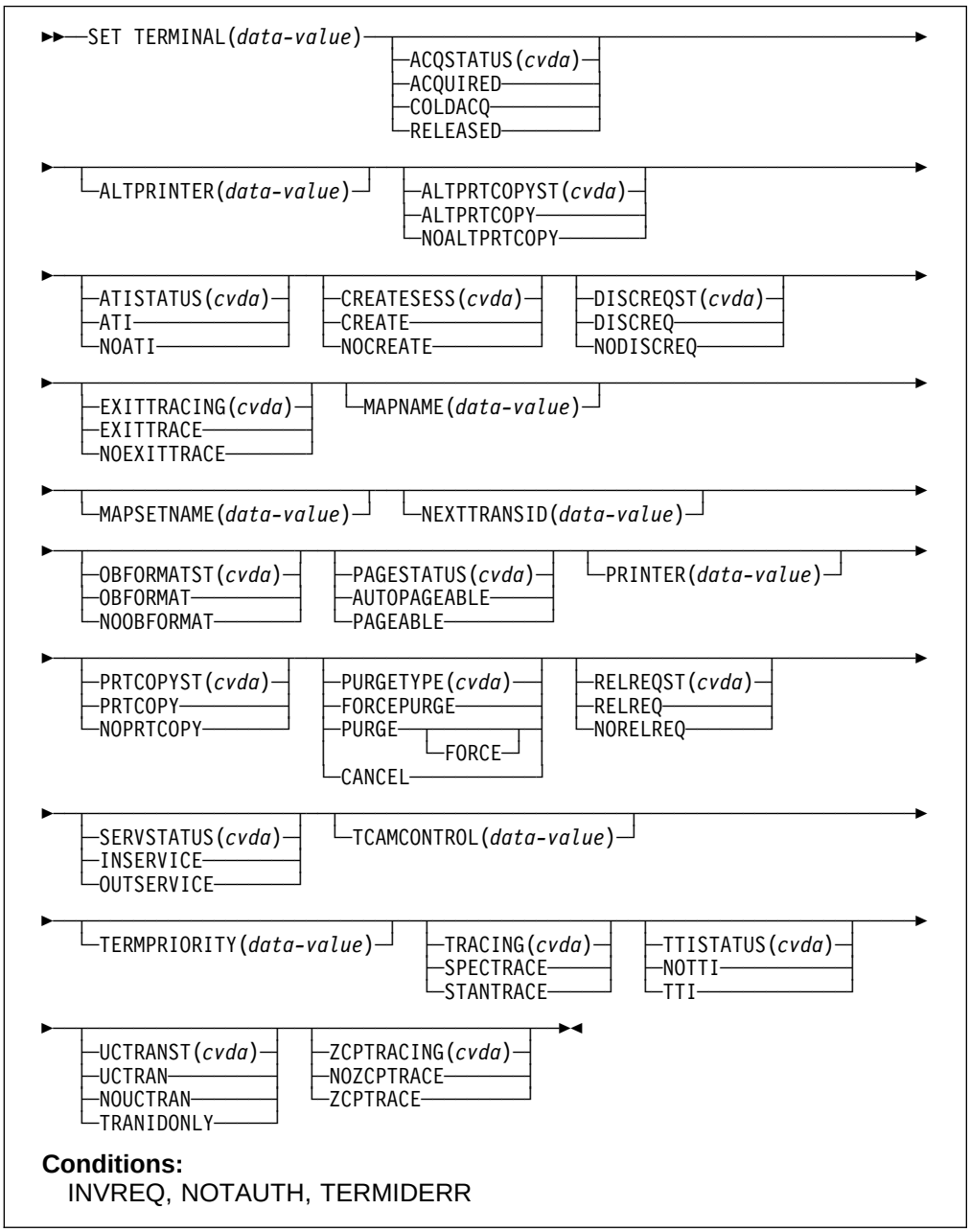
Syntax



For details of the keyword ATIUSERID see “EXEC CICS SET TDQUEUE command” on page 132.

SET TERMINAL

Syntax



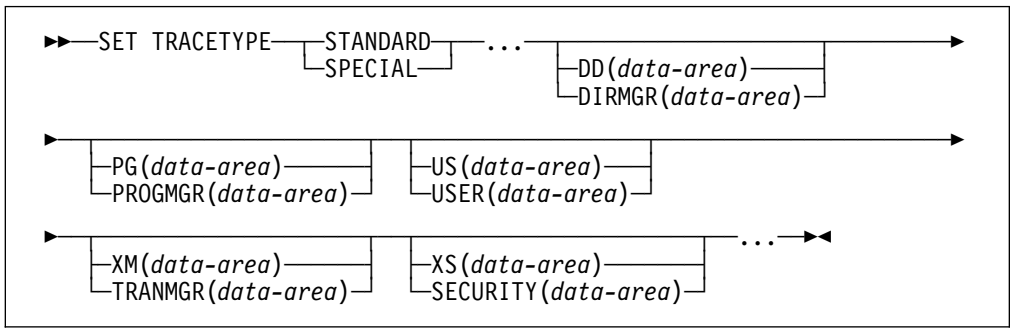
Conditions:
INVREQ, NOTAUTH, TERMIDERR

For details of the keyword PURGETYPE see “EXEC CICS SET TERMINAL command” on page 240.

For details of the keywords MAPNAME and MAPSETNAME, see “MAPNAME and MAPSETNAME options on INQUIRE and SET TERMINAL” on page 473.

SET TRACETYPE

Syntax



For details of the keywords DD and DIRMGR see “INQUIRE TRACETYPE command” on page 328.

For details of the keywords PG and PROGMGR see “EXEC CICS SET TRACETYPE command” on page 336.

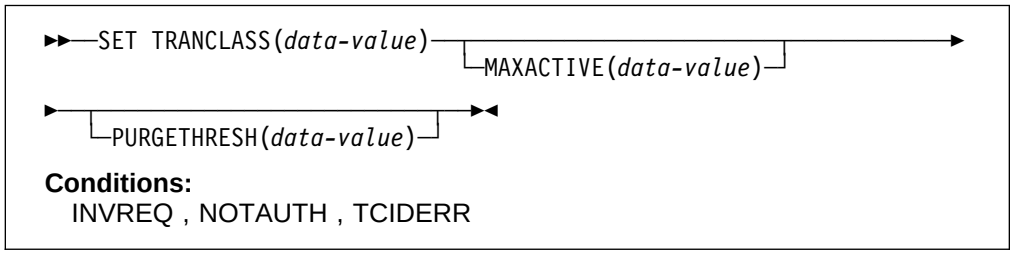
For details of the keywords US and USER see “Changes to the system programming interface” on page 405.

For details of the keywords XM and TRANMGR see “EXEC CICS INQUIRE and SET TRACETYPE command” on page 353.

For details of the keywords XS and SECURITY see “Changes to the system programming interface” on page 395.

SET TRANCLASS

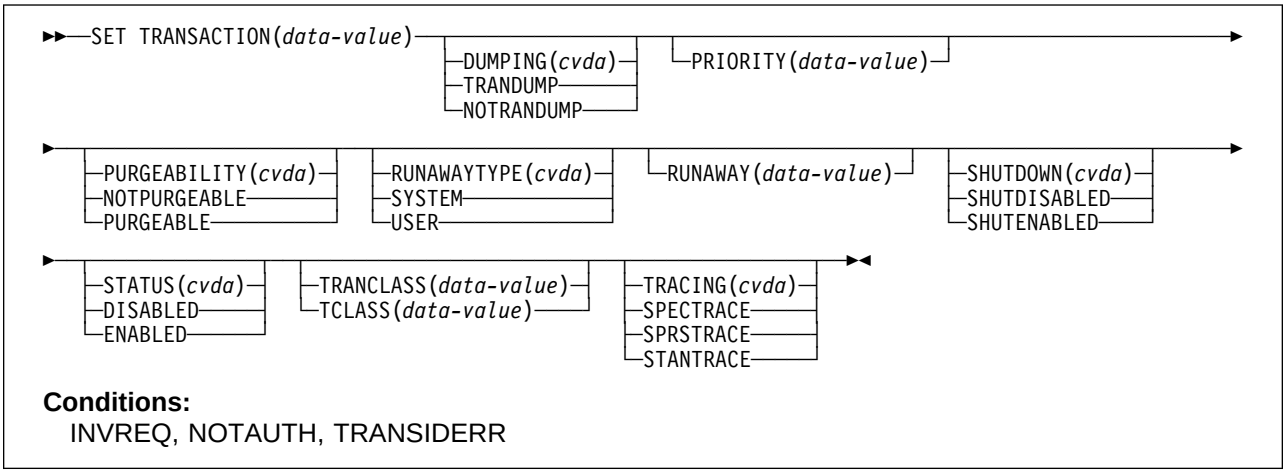
Syntax



For details of the keywords MAXACTIVE, and PURGETHRESH see “EXEC CICS SET TRANCLASS command” on page 359.

SET TRANSACTION

Syntax



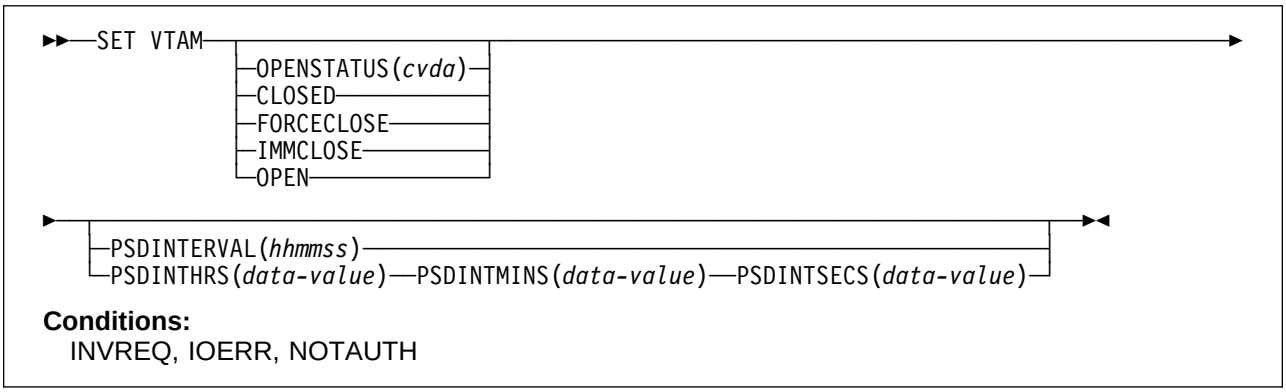
Conditions:
INVREQ, NOTAUTH, TRANSIDERR

For details of the keywords RUNAWAY, RUNAWAYTYPE, SHUTDOWN, TCLASS, and TRANSCLASS see “EXEC CICS SET TRANSACTION command” on page 356.

SET VTAM

Function Open/close the VTAM ACB and set the PSDI.

Syntax



Conditions:
INVREQ, IOERR, NOTAUTH

For details of the keywords PSDINTERVAL, PSDINTHRS, PSDINTMINS, PSDINTSECS see “EXEC CICS SET VTAM command” on page 63.

Chapter 36. Changes to the master terminal transaction (CEMT)

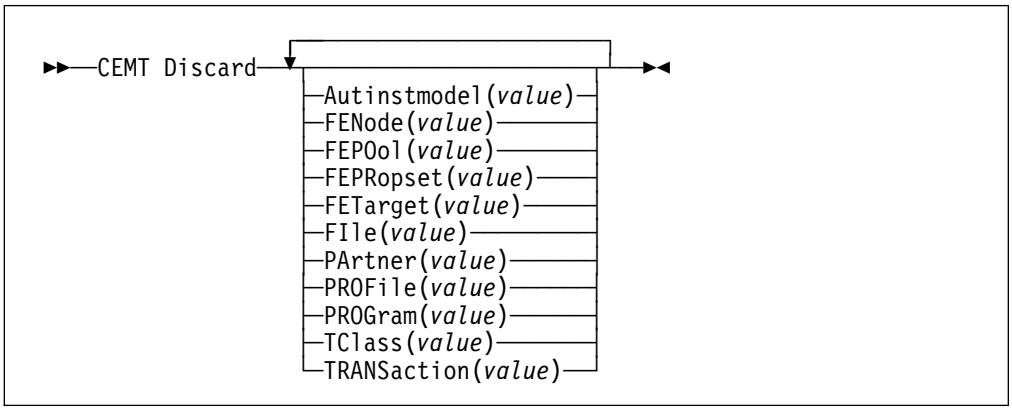
This chapter shows all the changes to the master terminal transaction (CEMT) made in CICS/ESA 4.1. The syntax diagrams show all the parameters, with the changed or new parameters marked.

CEMT DISCARD

Function

Remove an installed resource definition and its corresponding catalog entry from an active CICS system.

Command syntax



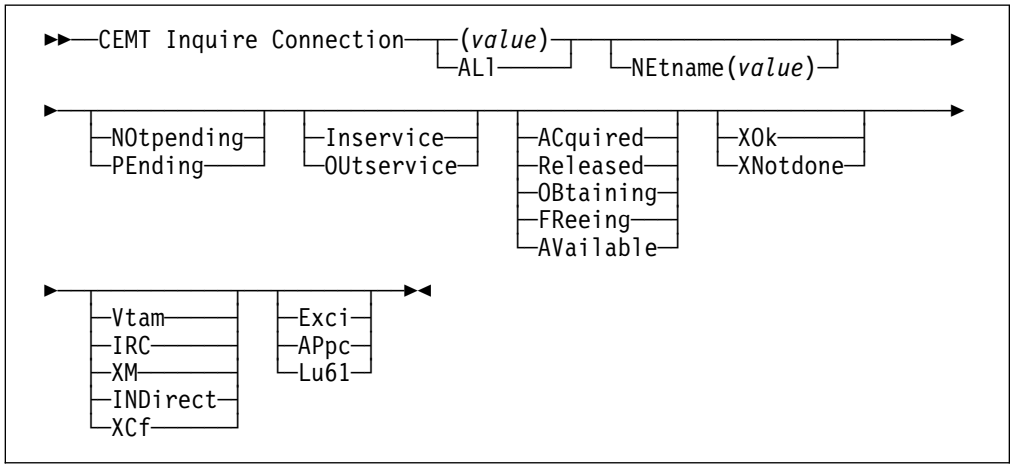
For information about the new TCLASS option, see Chapter 21, “Transaction manager domain” on page 341.

For information about the FEPI options FENODE, FEPOOL, FEPROPSET, and FETARGET, see the *CICS/ESA Front End Programming Interface User’s Guide*.

CEMT INQUIRE CONNECTION

Function
Retrieve information about system connections.

Command syntax



For information about the new EXCI option, see Chapter 11, “External CICS interface” on page 169.

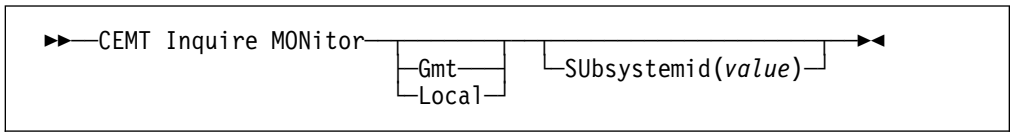
For information about the new XCF option, see Chapter 5, “Cross-system multiregion operation (XCF/MRO)” on page 87.

For information about the additional information provided for MRO connections, see the *CICS/ESA CICS-Supplied Transactions*.

CEMT INQUIRE MONITOR

Function
Retrieve information about the status of CICS monitoring.

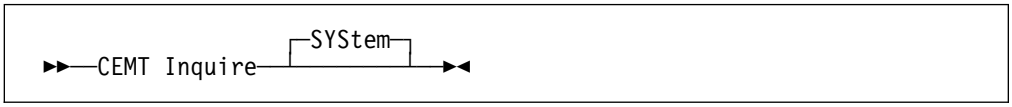
Command syntax



For information about the GMT, LOCAL and SUBSYSTEMID options, see Chapter 18, “Monitoring and statistics” on page 295.

CEMT INQUIRE SYSTEM

Command syntax



The INQUIRE SYSTEM command returns the following display:

```
IN SYSTEM
STATUS:  RESULTS - OVERTYPE TO MODIFY
  AGing( 32768 )                    RDsasize( 00262144 )
  AKp( 00200 )                      RElease( 0410 )
  CDsasize( 00524288 )              RUnaway( 0005000 )
  DFltuser( CICSUSER )              SCandelay( 0050 )
  DSalimit( 04194304 )              SDsasize( 00262144 )
  DTrprogram( DFHDYP   )            SOSAbove( NOTSOS )
  ECdsasize( 0002097152 )           SOSBelow( NOTSOS )
  EDsalimit( 0020971520 )           SToreprotect( INACTIVE )
  ERdsasize( 0003145728 )           TIme( 0000100 )
  ESdsasize( 0000000000 )           TRanisolate( INACTIVE )
  EUdsasize( 0001048576 )           Udsasize( 00000000 )
  MAxtasks( 032 )
  MRobatch( 001 )
  OPRel( 22 )
  OPSys( X )
  PROGAUTOCtlg( CTLGMODIFY )
  PROGAUTOExit( DFHPGADX )
  PROGAUTOInst( AUTOINACTIVE )
```

Figure 56. CEMT INQUIRE SYSTEM screen

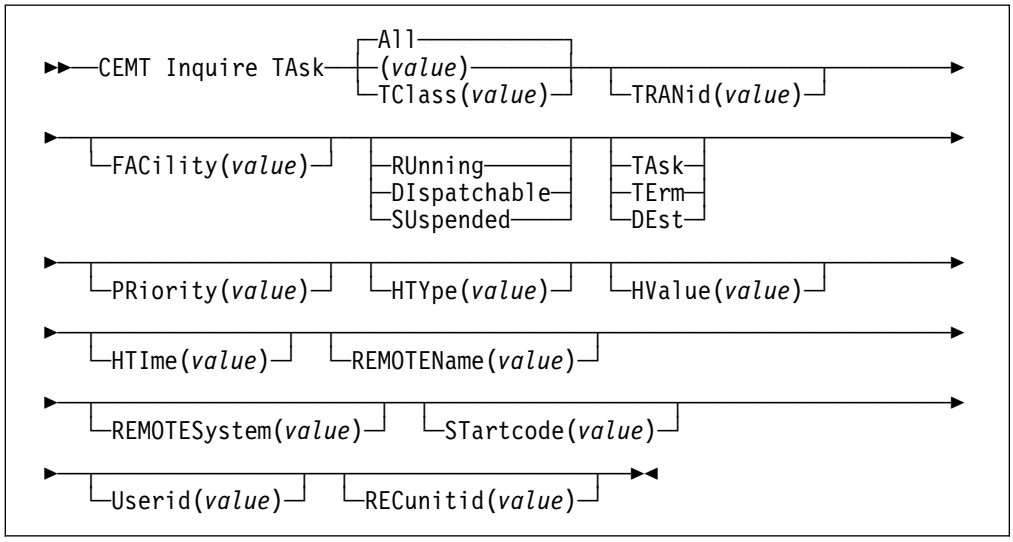
For information about CMDPROTECT, REENTPROTECT, DSALIMIT, EDSALIMIT, ESDSASIZE, RDSASIZE, SDSASIZE, and TRANISOLATE, see Chapter 2, “Transaction isolation” on page 9.

For information about PROGAUTOCTLG, PROGAUTOEXIT, and PROGAUTOINST, see Chapter 14, “Autoinstall for programs, mapsets, and partitionsets” on page 249.

CEMT INQUIRE TASK

Function
Retrieve information about a user task.

Command syntax

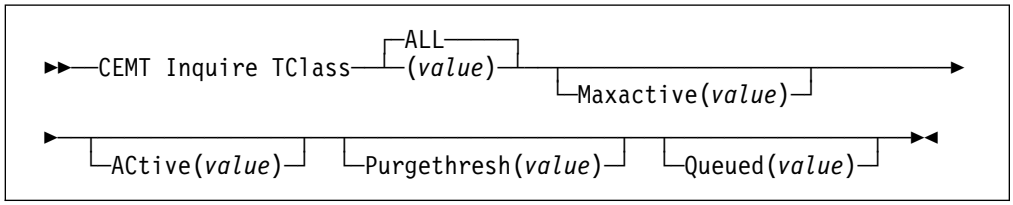


For information about the changes to INQUIRE TASK, see Chapter 21, “Transaction manager domain” on page 341.

CEMT INQUIRE TCLASS

Function
Retrieve information about the transaction classes installed.

Command syntax

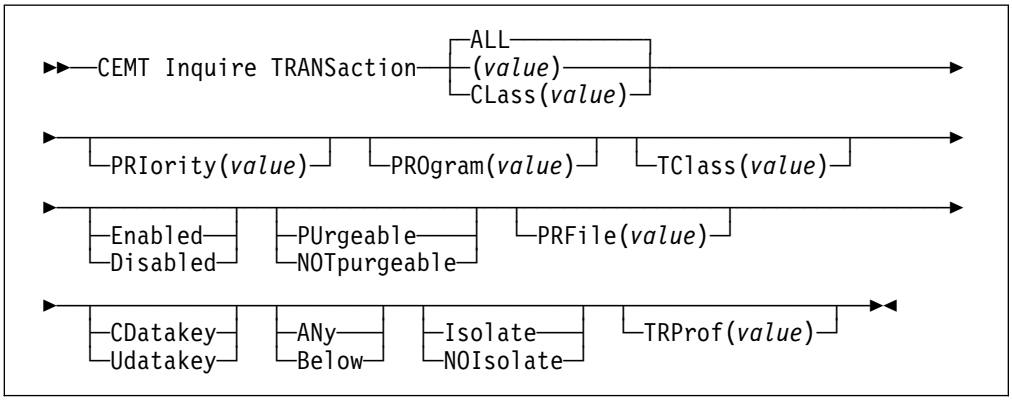


For information about the new options, see Chapter 21, “Transaction manager domain” on page 341.

CEMT INQUIRE TRANSACTION

Function
Retrieve information about transactions.

Command syntax

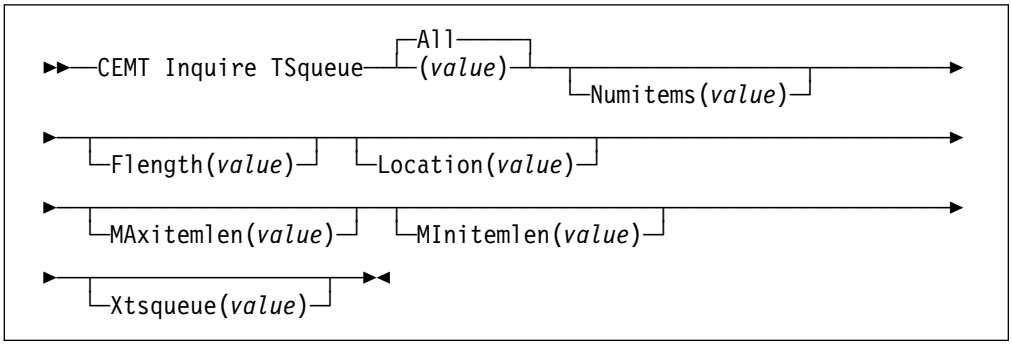


For information about the new TCLASS option, see Chapter 21, “Transaction manager domain” on page 341

CEMT INQUIRE TSQUEUE

Function
Retrieve information about temporary storage queues.

Command syntax



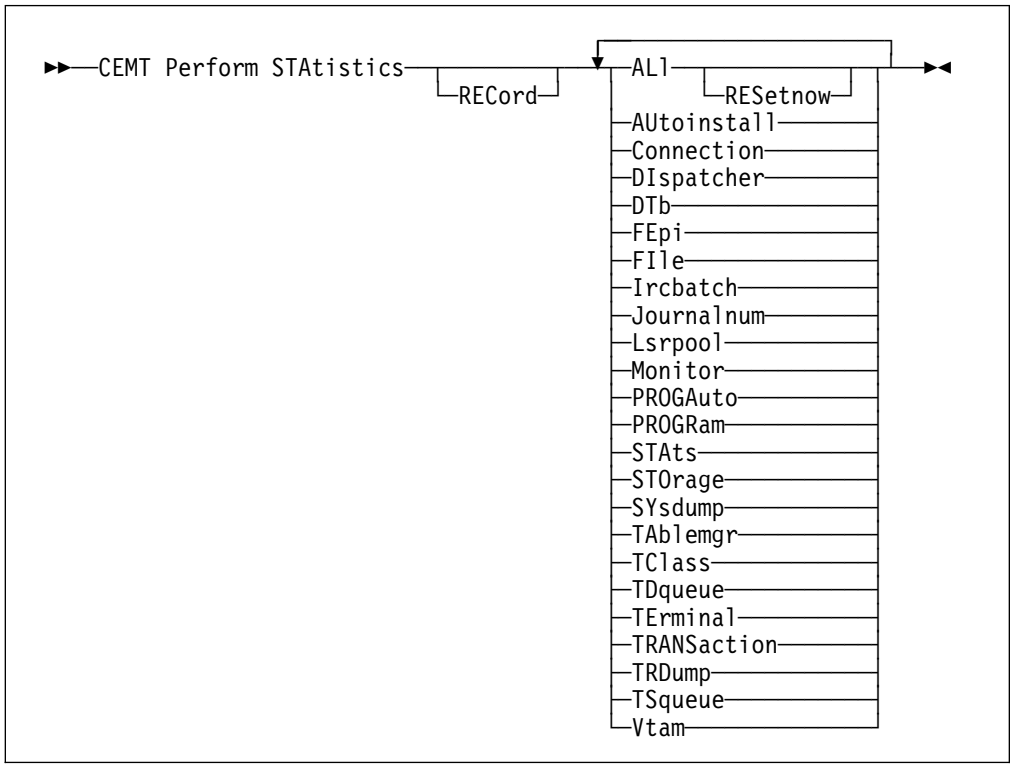
For information about the options on INQUIRE TSQUEUE, see “New CEMT command to inquire on temporary storage queues” on page 534.

CEMT PERFORM STATISTICS

Function

Write the statistics for a named resource type immediately to the SMF data set, rather than wait for the current statistics-gathering interval to expire.

Command syntax

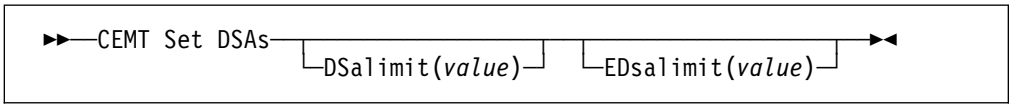


For information about the new options, see Chapter 18, “Monitoring and statistics” on page 295.

CEMT SET DSAS

Function
Change the system storage attributes.

Command syntax

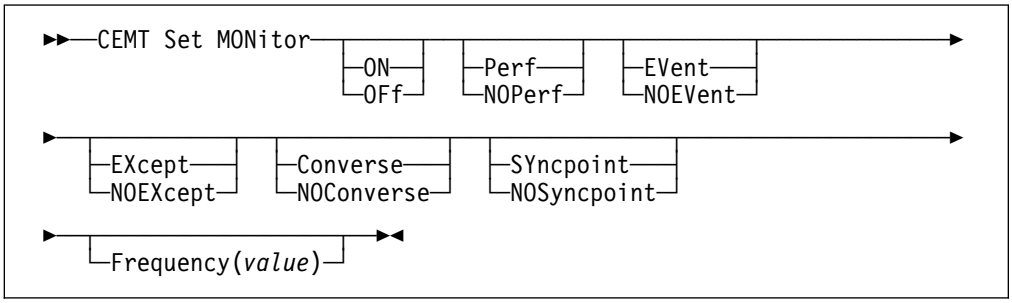


For information about the new options, see Chapter 2, “Transaction isolation” on page 9.

CEMT SET MONITOR

Function
Change the status of monitoring.

Command syntax



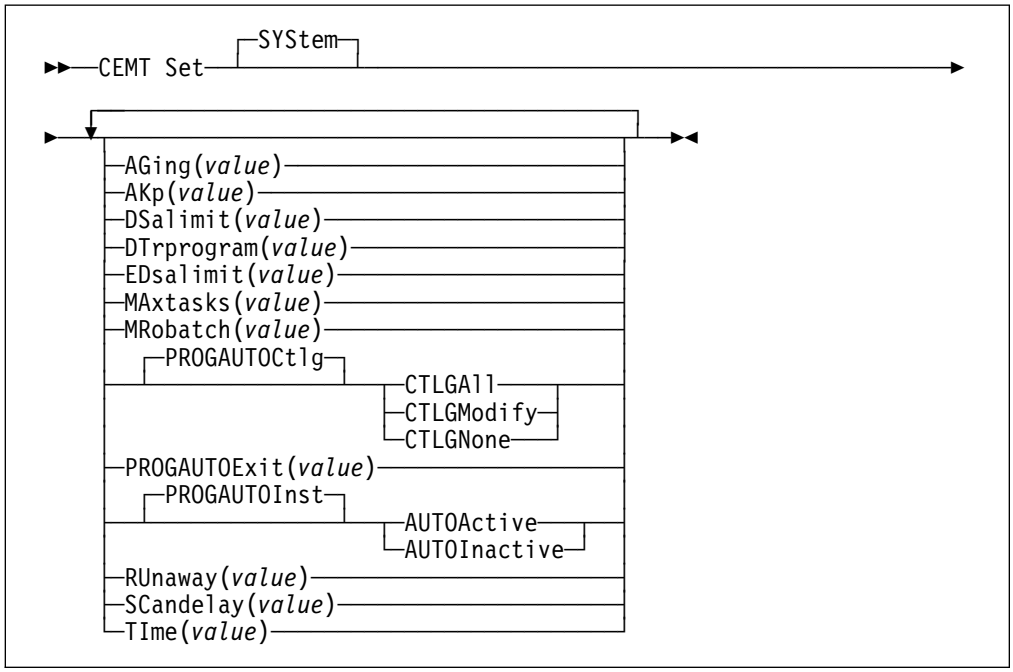
For information about the new options, see Chapter 18, “Monitoring and statistics” on page 295.

CEMT SET SYSTEM

Function

Change the system attributes.

Command syntax



For information about the DSALIMIT and EDSALIMIT options, on both INQUIRE and SET SYSTEM commands, see Chapter 2, “Transaction isolation” on page 9.

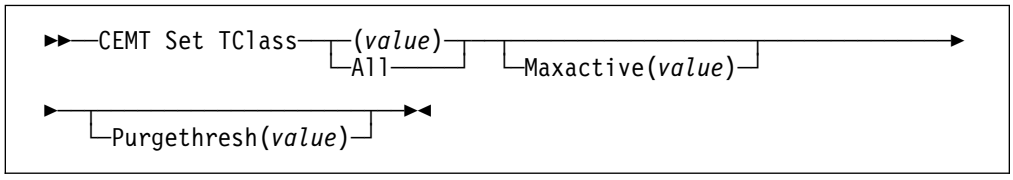
For information about the PROGAUTOCTLG, PROGAUTOEXIT, and PROGAUTOINST options, on both INQUIRE and SET SYSTEM commands, see Chapter 14, “Autoinstall for programs, mapsets, and partitionsets” on page 249.

CEMT SET TCLASS

Function

Reset the maximum number of tasks and the purge threshold for a transaction class.

Command syntax



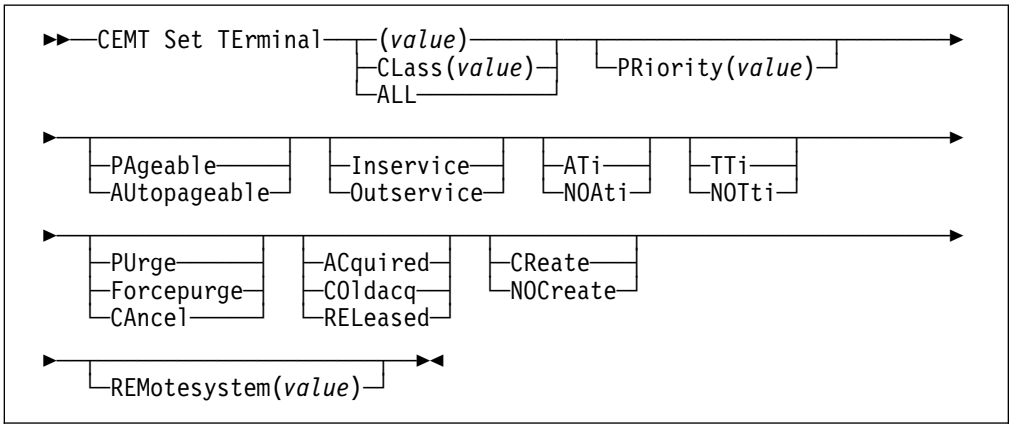
For information about the new options, see Chapter 21, “Transaction manager domain” on page 341.

CEMT SET TERMINAL

Function

Change the attributes of named terminals.

Command syntax



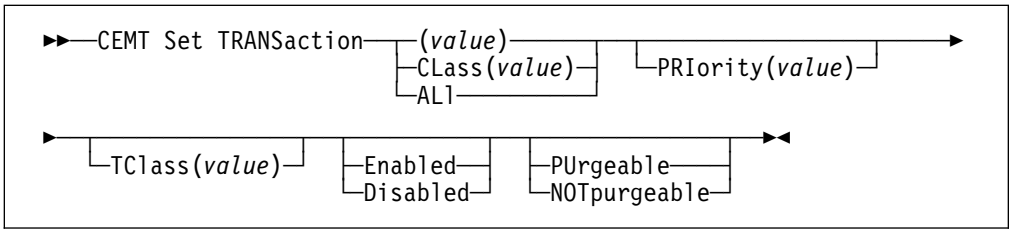
For information about the CANCEL option, see Chapter 13, “Cancel start requests” on page 239.

CEMT SET TRANSACTION

Function

Change some of the attributes of a selected transaction.

Command syntax



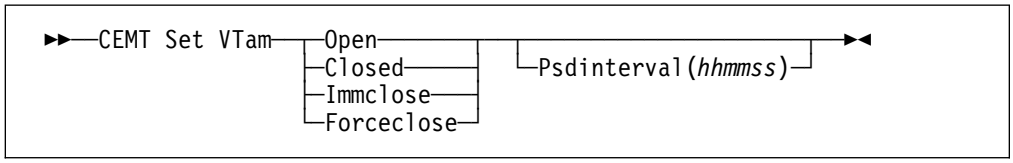
For information about the TCLASS option, see Chapter 21, “Transaction manager domain” on page 341.

CEMT SET VTAM

Function

Open or close the VTAM ACB and set the persistent session delay interval (PSDI).

Command syntax



For information about the PSDINTERVAL options, see Chapter 3, “VTAM persistent sessions” on page 51.

Chapter 37. Changes to resource definition

This chapter lists all changes and additions to the resource definition interface.

The changes to resource definition online (RDO) are:

- There are five new attributes on the TRANSACTION definition:
  - STORAGECLEAR
  - RUNAWAY
  - SHUTDOWN
  - TRANCLASS
  - ISOLATE.
- The TCLASS attribute on the TRANSACTION definition is now obsolete.
- A new resource type, TRANCLASS, has been added.
- The meaning of EXECKEY on the PROGRAM definition has been changed.
- There are new attributes on the CONNECTION definition:
  - CONNTYPE
  - PSRECOVERY
  - QUEUELIMIT
  - REMOTESYSNET
  - USEDFLTUSER (added by APAR PN 63960/PTF UN77613)
- There are new attributes on the TERMINAL definition:
  - REMOTESYSNET
  - USEDFLTUSER
- A new value, EXCI, on the PROTOCOL attribute of the CONNECTION definition.
- A new value, EXCI, on the PROTOCOL attribute of the SESSIONS definition.
- Changes to the RECOVNOTIFY and RECOVOPTION attributes of the SESSIONS definition.
- Changes to the RECOVNOTIFY and RECOVOPTION attributes of the TYPETERM definition.
- CEDA gives the current resource name for the ALTER, DEFINE, and VIEW commands.
- Before-images of a CEDA ALTER command are logged to the CSDL queue.
- New and changed CICS-supplied resource definitions.

The changes to resource definition at macro level are:

- CPU and CONV are removed from DFHMCT TYPE=INITIAL
- Increase in size of the MOVE option of the PERFORM operand of DFHMCT TYPE=EMP
- Removal of the DFHPCT macro table
- Removal of the DFHPPT macro table
- Removal of VSAM file definition from DFHFCT
- Removal of data tables definition from DFHFCT
- Removal of local shared resource pool definition from DFHFCT.

Changes to the PROGRAM definition

As a result of the introduction of transaction isolation, the meaning of the EXECKEY attribute has changed. See page 25 for a description of EXECKEY.

Changes to the TRANSACTION definition

There are five new attributes on the TRANSACTION resource definition;

- ISOLATE
- RUNAWAY
- SHUTDOWN
- STORAGECLEAR
- TRANCLASS

As a result of the introduction of the ISOLATE option, the description of TASKDATAKEY has been changed.

As a result of the introduction of TRANCLASS, the TCLASS attribute is obsolete in CICS/ESA 4.1. If you already use TCLASS, you can continue to use it in existing definitions, provided that you install the corresponding TRANCLASS resource definitions. See the *CICS/ESA Migration Guide* for information about TRANCLASS definitions that correspond to the old TCLASS numbers.

See Figure 57 on page 617 for full details of the CEDA define transaction panel.

ISOLATE
See page 23 for a description of ISOLATE.

RUNAWAY
See page 345 for a description of RUNAWAY.

SHUTDOWN
See page 345 for a description of SHUTDOWN.

STORAGECLEAR description
See page 345 for a description of STORAGECLEAR.

TASKDATAKEY description
See page 24 for a description of TASKDATAKEY.

TRANCLASS description
See page 345 for a description of TRANCLASS.

```
 TRansaction  ==>
 Group        ==>
 DEscription  ==>
 PROGram      ==>
 TWasize      ==> 00000             0-32767
 PROFile      ==> DFHCICST
 PArtitionset ==>
 STAtus       ==> Enabled           Enabled | Disabled
 PRIMedsize     : 00000             0-65520
 TASKDATALoc  ==> Below             Below | Any
 TASKDATAKey  ==> User              User | Cics
 STOrageCLear ==> No                No | Yes
 RUnaway      ==> System            System | 0-2700000
 SHutdown     ==> Disabled          Disabled | Enabled
 ISolate      ==> Yes               Yes | No
REMOTE ATTRIBUTES
 DYnamic      ==> No                No | Yes
 REMOTESystem ==>
 REMOTEName   ==>
 TRProf       ==>
 Localq       ==>                   No | Yes
SCHEDULING
 PRIOrity     ==> 001               0-255
 TClass         : No                No | 1-10
 TRANCLASS    ==> DFHTLC00
ALIASES
 Alias        ==>
 TASKReq      ==>
 XTRanid      ==>
 TPName       ==>
              ==>
 XTPname      ==>
              ==>
              ==>

RECOVERY
 DTimout      ==> No                No | 1-6800
 INdoubt      ==> Backout           Backout | Commit | Wait
 RESTart      ==> No                No | Yes
 SPurge       ==> No                No | Yes
 TPUrge       ==> No                No | Yes
 DUmp         ==> Yes               Yes | No
 TRACe        ==> Yes               Yes | No
SECURITY
 RESSec       ==> No                No | Yes
 Cmdsec       ==> No                No | Yes
 Extsec         : No
 TRANSec        : 01                1-64
 RSl            : 00                0-24 | Public
```

Figure 57. The DEFINE panel for TRANSACTION

Changes to the CONNECTION definition

There are new attributes and values on the CONNECTION definition:

- CONNTYPE
- EXCI (new value on the PROTOCOL attribute)
- PSRECOVERY
- QUEUELIMIT
- REMOTESYSNET
- USEDFLTUSER (added by APAR PN 63960/PTF UN77613)

In addition to these new attributes, the meaning of the REMOTESYSTEM option has changed, and the BINDPASSWORD option is obsolete. BINDPASSWORD continues to be supported in compatibility mode only.

```
 Connection     :
 Group          :
 DEscription  ==>
CONNECTION IDENTIFIERS
 Netname      ==>
 INDsys       ==>
REMOTE ATTRIBUTES
 REMOTESystem ==>
 REMOTESYsnet ==>
 REMOTEName   ==>
CONNECTION PROPERTIES
 ACcessmethod ==> Vtam              Vtam | IRc | INdirect | Xm
 PRotocol     ==> Appc              Appc | Lu61 | Exci
 Conntype     ==>                   Generic | Specific
 SInglesess   ==> No                No | Yes
 DAtastream   ==> User              User | 3270 | SCs | STrfield | Lms
 RECordformat ==> U                 U | Vb
 Queuelimit   ==> No                No | 0-9999
 Maxqtime     ==> No                No | 0-9999
OPERATIONAL PROPERTIES
 AUtoconnect  ==> No                No | Yes | All
 INService    ==> Yes               Yes | No
SECURITY
 SEcurityname ==>
 ATtachsec    ==> Local             Local | Identify | Verify | Persistent
                                    | Mixidpe
 BINDPassword ==:                   PASSWORD NOT SPECIFIED
 BINDSecurity ==> No                No | Yes
 USEDFLTUSER  ==> No                No | Yes

RECOVERY
 PSrecovery   ==> Sysdefault        Sysdefault | None
```

Figure 58. CEDA DEFINE CONNECTION panel

CONNTYPE
See page 200 for a description of CONNTYPE.

PROTOCOL
See page 200 for a description of PROTOCOL.

PSRECOVERY
See page 59 for a description of PSRECOVERY.

QUEUELIMIT and MAXQTIME
See page 73 for a description of QUEUELIMIT.

REMOTESYSNET
See page 502 for a description of REMOTESYSNET.

REMOTESYSTEM
See page 503 for a description of REMOTESYSTEM.

USEDFLTUSER
See the *CICS/ESA CICS-RACF Security Guide* for information about using the USEDFLTUSER option and the effect on FMH binds to other CICS releases and platforms. See also Table 40 on page 479.

Changes to the SESSIONS definition

There is a new value (EXCI) on the PROTOCOL attribute, shown in Figure 59. The description of SENDCOUNT has been changed as a result of the introduction of EXCI; the new description is on page 201.

The changes to the RECOVOPTION and RECOVNOTIFY attributes for SESSIONS are identical to those for the TYPETERM definition; see “Changes to the TYPETERM definition” on page 620 for the descriptions of these attributes.

```
 Sessions       :
 Group          :
 DEscription  ==>
SESSION IDENTIFIERS
 Connection   ==>
 SESSName     ==>
 NETnameq     ==>
 MOdename     ==>
SESSION PROPERTIES
 Protocol     ==> Appc              Appc | Lu61 | Exci
 MAximum      ==> 001 , 000         0-999
 RECEIVEPfx   ==>
 RECEIVECount ==>                   1-999
 SENDPfx      ==>
 SENDCount    ==>                   0-999
 SENDSize     ==> 04096             1-30720
 RECEIVESize  ==> 04096             1-30720
 SESSPriority ==> 000               0-255
 Transaction    :
OPERATOR DEFAULTS
 OPERId         :
 OPERPriority   : 000               0-255
 OPERRsl        : 0
 OPERSecurity   : 1
PRESET SECURITY
 USERId       ==>
OPERATIONAL PROPERTIES
 Autoconnect  ==> No                No | Yes | All
 INservice      :
 Buildchain   ==> Yes               Yes | No
 USERArealen  ==> 000               0-255
 IOarealen    ==> 00000 , 00000     0-32767
 RELreq       ==> No                No | Yes
 DIscreq      ==> No                No | Yes
 NEPclass     ==> 000               0-255
RECOVERY
 RECOVOption  ==> Sysdefault        Sysdefault | Clearconv | Releasesess
                                    | Uncondrel | None
 RECOVNotify  ==> None              None | Message | Transaction
```

Figure 59. The DEFINE panel for SESSIONS

PROTOCOL description

See page 201 for a description of PROTOCOL.

SENDCOUNT description

See page 201 for the changed description of SENDCOUNT.

Changes to the TERMINAL definition

There is a new attribute on the TERMINAL resource definition:

- REMOTESYSNET
- USEDFLTUSER

In addition, the meaning of the REMOTESYSTEM option has changed.

REMOTESYSNET
See page 502 for a description of REMOTESYSNET.

REMOTESYSTEM
See page 502 for a description of REMOTESYSTEM.

USEDFLTUSER (APAR PN 63960/PTF UN77613)
See the *CICS/ESA CICS-RACF Security Guide* for information about using the USEDFLTUSER option and the effect on FMH binds to other CICS releases and platforms. See also Table 40 on page 479.

Changes to the TYPETERM definition

Prior to CICS/ESA 4.1, the RECOVNOTIFY attribute applied to class 1 terminals only, and the RECOVOPTION attribute applied to class 1 and class 2 terminals, with both options having effect only on an XRF takeover. In CICS/ESA 4.1, they are available to all terminal types (except LU0 pipeline and LU6.1 sessions), and take effect for persistent session support and XRF.

RECOVNOTIFY description
See page 57 for a description of RECOVNOTIFY.

RECOVOPTION description
See page 60 for a description of RECOVOPTION.

The new TRANCLASS definition

The TRANCLASS definition allows you to define a transaction class. By assigning transactions to transaction classes, you have more control over how CICS dispatches tasks. See Figure 60 for an example of a DEFINE TRANCLASS panel displayed in response to a CEDA DEFINE TRANCLASS(XXXX) GROUP(MYGROUP) MAXACTIVE(10) command.

```
CEDA  DEFine TRANClass( XXXX     ) GROUP(MYGROUP)
 TRANClass      : XXXX
 Group          : MYGROUP
 Description  ==>
CLASS LIMITS
 Maxactive    ==> 010               0-999
 Purgethresh  ==> No                No | 1-1000000
```

Figure 60. CEDA DEFINE TRANCLASS panel

GROUP attribute
See page 346 for a description of GROUP.

MAXACTIVE attribute
See page 346 for a description of MAXACTIVE.

PURGETHRESH attribute
See page 346 for a description of PURGETHRESH.

TRANCLASS attribute
See page 347 for a description of TRANCLASS.

Logging of before-images of the ALTER command

In CICS/ESA 3.3, when the ALTER command is used to change a resource, its after-image is logged to the CSDL transient data queue. In CICS/ESA 4.1, the before-image is also logged to CSDL.

The audit record written to CSDL is a copy of the ALTER command, with the before-image and the after-image identified by 'BEFORE' and 'AFTER'. Figure 61 shows three audit records, one for a DEFINE command, and two associated with an ALTER command.

```
CEDA CICSUSER 93.154 16.13.18 DEFINE MAPSET(MAP1) GROUP(MYGROUP) RESIDENT(NO)
                              USAGE(NORMAL) USELPACOPY(NO) STATUS(ENABLED) RSL(0)
CEDA CICSUSER BEFORE 93.154 16.13.18 ALTER MAPSET(MAP1) GROUP(MYGROUP) RESIDENT(NO)
                                     USAGE(NORMAL) USELPACOPY(NO) STATUS(ENABLED) RSL(0)
CEDA CICSUSER AFTER  93.154 16.17.39 ALTER MAPSET(MAP1) GROUP(MYGROUP) RESIDENT(NO)
                                     USAGE(TRANSIENT) USELPACOPY(NO) STATUS(ENABLED) RSL(0)
```

Figure 61. Logging of before- and after-image of ALTER command

Provision of resource name in the ALTER, DEFINE, and VIEW commands

When modifying resources using CEDA, most resources have more than one screen full of attributes. In CICS/ESA 3.3, the resource name was provided on the first screen only.

In CICS/ESA 4.1, the resource name is provided on every screen. Figure 62 on page 622 shows the second screen (on a 24-line display) of a CEDA ALTER TRANSACTION(AAAA) GROUP(MYGROUP) command, with the CEDA command repeated on the second line.

```
 OBJECT CHARACTERISTICS
  CEDA  ALter TRANSaction( AAAA )
+  PRIMedsize     : 00000             0-65520
   TASKDATALoc  ==> Below             Below | Any
   TASKDATAKey  ==> User              User | Cics
   STOrageCLear ==> No                No | Yes
   RUnaway      ==> System            System | 0-2700000
   SHutdown     ==> Disabled          Disabled | Enabled
   ISolate      ==> Yes               Yes | No
  REMOTE ATTRIBUTES
   DYnamic      ==> No                No | Yes
   REMOTESystem ==>
   REMOTEName   ==>
   TRProf       ==>
   Localq       ==>                   No | Yes
  SCHEDULING
   PRIOrity     ==> 001               0-255
   TClass         : No                No | 1-10
+  TRANCLASS    ==> DFHTLC00

PF 1 HELP 2 COM 3 END       6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 62. Second screen of a CEDA ALTER TRANSACTION command

New and changed CICS-supplied resource definitions

The changed resource definitions are:

- DFHCRP is removed from the CICS-supplied resource definitions.

There are five new supplied resource definitions:

- Default program definition DFHPGAPG

```
GROUP(DFHPGAIP)       PROGRAM(DFHPGAPG)
DESCRIPTION(Default model program definition for program autoinstall)
                      RELOAD(NO)            RESIDENT(NO)
USAGE(NORMAL)         USELPACOPY(NO)        STATUS(ENABLED)
CEDF(YES)             DATALOCATION(BELOW)   EXECKEY(USER)
EXECUTIONSET(FULLAPI)
```

  Although this default program resource definition does not specify a program language, the CICS autoinstall routine detects the program language from the program being loaded, and sets the language field accordingly.

- Default mapset definition DFHPGAMP

```
GROUP(DFHPGAIP)       MAPSET(DFHPGAMP)
DESCRIPTION(Default model mapset definition for program autoinstall)
RESIDENT(NO)    USAGE(NORMAL)    USELPACOPY(NO)    STATUS(ENABLED)
```

- Default partitionset definition DFHPGAPT

```
GROUP(DFHPGAIP)     PARTITIONSET(DFHPGAPT)
DESCRIPTION(Default model partitionset defn for program autoinstall)
RESIDENT(NO)     USAGE(NORMAL)     USELPACOPY(NO)    STATUS(ENABLED)
```

```
DEFINE PROFILE(DFHPPF01)
       DESCRIPTION(VTAM only.  For tasks attached before CSD defns)
       GROUP(DFHSTAND)
       SCRNSIZE(DEFAULT)
                               MSGJRNL(NO)
       MSGINTEG(NO)            ONEWTE(NO)              PROTECT(NO)
       DVSUPRT(VTAM)           INBFMH(NO)              RAQ(NO)
       LOGREC(NO)
       NEPCLASS(000)

DEFINE PROFILE(DFHPPF02)
       DESCRIPTION(all nulls.  For tasks attached before CSD defns)
       GROUP(DFHSTAND)
       SCRNSIZE(DEFAULT)
                               MSGJRNL(NO)
       MSGINTEG(NO)            ONEWTE(NO)              PROTECT(NO)
       DVSUPRT(ALL)            INBFMH(NO)              RAQ(NO)
       LOGREC(NO)
       NEPCLASS(000)
```

Warning: If you are migrating from any release of CICS earlier than CICS/ESA 3.3, and you are not using autoinstall for programs, you should ensure that the CICS-supplied group, DFHEDP, introduced in CICS/ESA 3.3, is added your start-up group list. This contains the program definition for the EXEC DL/I translation program, and is needed for both the CICS local DL/I and the CICS-DBCTL environments.

Note: This does not apply if the program autoinstall exit program (PGAIEXIT=DFHPGADX, as specified in the SIT) is active, because DFHEDP will be installed automatically.

Changes to macro-level resource definition

The changes to macro-level resource definition in CICS/ESA 4.1 are:

- removal of the DFHPCT macro table
- removal of the DFHPPT macro table
- removal of VSAM file definition from DFHFCT
- removal of data tables definition from DFHFCT
- removal of local shared resource pool definition from DFHFCT.

Changes to DFHMCT

The CPU and CONV parameters of the DFHMCT TYPE=INITIAL are removed. The CONV parameter is now specified as a system initialization parameter. The CPU option is no longer required because CPU time is always measured. The TYPE=INITIAL macro for the MCT is as follows:

```
DFHMCT TYPE=INITIAL
      [,SUFFIX=xx]
```

DFHMCT TYPE=EMP

The MOVE option of the PERFORM operand of the DFHMCT TYPE=EMP macro definition is enhanced to increase the maximum size of user character fields from 256 bytes to 8192 bytes. These enhancements are as follows:

PERFORM=(option|,...|)

This operand must be coded when CLASS=PERFORM is coded. It specifies that information is to be added to or changed in the user fields of a performance class data record by this user event monitoring point (EMP). The user fields for each user are distinguished by a separate name in the ID operand and can comprise:

- Up to 256 counters
- Up to 256 clocks, each made up of a 4-byte accumulator and 4-byte count
- A byte string of up to 8192 bytes.

Note: The maximum size of user data in each performance record is increased from 4096 bytes to 16384 bytes.

Actions will be performed on the user fields according to the options specified. PERFORM can be abbreviated to PER.

Changed options for the PERFORM operand are:

⋮

MOVE(number3,number4)

A string of data is to be moved into the user byte-string field. To use this option, both the DATA1 and DATA2 fields must be passed from the user EMP.

The user byte-string field is updated starting at the offset specified by number3. The data to be moved starts at the address supplied in the DATA1 field. The maximum length of data that can be moved is given by number4 (in bytes), and the actual length of data that is to be moved is given by the value of the DATA2 field. If the value of DATA2 is zero, then the length of the data given by number4 is moved.

Number3 is a decimal integer in the range 0 to 8191, and number4 is a decimal integer in the range 1 to 8192. The maximum length of the user character field is (number3 + number4), and must be in the range 1 to 8192.

Note: Only one of the MLTCNT and MOVE options can be used in each DFHMCT TYPE=EMP macro instruction.

⋮

DFHMCT TYPE=RECORD

DFHFEPI is introduced as a group field name to enable you to control the inclusion or exclusion of FEPI monitoring data as a group. Within the DFHFEPI group are FEPI field numbers to identify the individual FEPI fields.

DFHMCT	TYPE=RECORD ,CLASS=PERFORM ,EXCLUDE=(DFHFEPI)

Non-shipment of macros

The following macros are not shipped with CICS/ESA 4.1:

- The program control table, DFHPCT
- The processing program table, DFHPPT
- The user signon table, DFHSNT.

DFHPCT and DFHPPT: These tables have not been supported since CICS/ESA 3.1.1, and the macros were shipped with CICS/ESA 3.3 for migration purposes only. If you currently use the DFHPCT and DFHPPT macros to create your transaction and program definitions, you must migrate them to the CSD using the DFHCSDUP MIGRATE facility of your current release. There is no provision in CICS/ESA 4.1 for migrating the PCT and PPT. See the *Resource Definition Guide* for information on migrating from macro tables to the CSD.

DFHSNT: CICS does not support a signon table for user data in CICS/ESA 4.1. All user data is obtained from the external security manager—from the CICS segment where the ESM is RACF. You can migrate your existing signon table using the DFHSNMIG migration utility program.

Changes to DFHFCT

In CICS/ESA 4.1, you can no longer define the following resources using DFHFCT:

- VSAM files
- Data tables
- Local shared resource (LSR) pools.

The DFHFCT macro is retained in CICS/ESA 4.1 to allow FCT macro definitions to be migrated to the CSD, but when you do a CICS cold start only remote files (VSAM and BDAM) and local BDAM files are installed from the FCT. You should define all other resources in the CICS system definition data set (CSD).

If you currently use DFHFCT to create VSAM files, data tables, or LSR pools, you must migrate them to the CSD before using CICS/ESA 4.1. See the *Resource Definition Guide* for information on migrating macro-defined tables to the CSD.

Chapter 38. Changes to system definition

This chapter lists all changes and additions to CICS system initialization parameters

Table 57 lists those CICS system initialization parameters that have changed, been removed, or been added in CICS/ESA 4.1. Descriptions of the changed and new system initialization parameters are given after the table, starting on page 629.

Table 57 (Page 1 of 2). System initialization parameters changed or removed in CICS/ESA 4.1

	DFHSIT	[TYPE={**CSECT**|DSECT}]
		⋮
		~~[,AMXT={**MXT-number**|number}]~~
		⋮
|		[,CMDSEC={**ASIS**|ALWAYS}]
#		[,CDSASZE={**0K**|number}]
		⋮
		⋮
|		[,CMDPROT={**YES**|NO}]
		⋮
		~~[,CMXT=([n1][,n2]...)]~~
		~~[,CMXTLIM=([n1][,n2]...,n10)]~~
		⋮
|		[,CONFDATA={**SHOW**|HIDETC}]
|		[,CONFTXT={**NO**|YES}]
		⋮
		~~[,CSCS={**64K**|number}]~~
		⋮
|		[,DSALIM={**5M**|number}]
		⋮
|		[,DSHIPIDL={**020000**|hhmmss}]
|		[,DSHIPINT={**120000**|hhmmss}]
		⋮
|		[,DTRTRAN={**CRTX**|name}]
		⋮
#		[,ECDSASZE={**0M**|number}]
		~~[,ECSCS={**256K**|number}]~~
|		[,EDSALIM={**20M**|number}]
		⋮
#		[,ERDSASZE={**0M**|number}]
#		[,ESDSASZE={**0M**|number}]
		~~[,ERSCS={**256K**|number}]~~
		⋮
#		[,EUDSASZE={**0M**|number}]
		~~[,EUSCS={**256K**|number}]~~
		⋮
|		[,GNTRAN={**CESF**|transaction-id}]
		⋮
|		[,GRNAME=name]
|		[,GRPLIST={**DFHLIST**|name|
|		(name[,name2][,name3][,name4])}]

Table 57 (Page 2 of 2). System initialization parameters changed or removed in CICS/ESA 4.1

		⋮
		[,ICVR={**5000**|number}]
		⋮
		~~[,ISRDELAY={**30**|number}]~~
		⋮
		~~.[,MAXSMIR={**999**|number}]~~
		⋮
		[,MNCONV={**NO**|YES}]
		[,MNFREQ={**0**|hhmmss}]
		[,MNSUBSYS={**null**|xxxx}]
		[,MNSYNC={**NO**|YES}]
		[,MNTIME={**GMT**|LOCAL}]
		⋮
		[,MXT={**5**|number}]
		⋮
		[,PGAICTLG={**MODIFY**|NONE|ALL}]
		[,PGAIEXIT={**DFHPGADX**|name}]
		[,PGAIPGM={**INACTIVE**|ACTIVE}]
		⋮
		[,PLTPISEC={**NONE**|CMDSEC|RESSEC|ALL}]
		[,PLTPIUSR=userid]
		⋮
		[,PSDINT={**0**|hhmmss}]
		⋮
+		[,RDSASZE={**0K**|number}]
		[,RESSEC={**ASIS**|ALWAYS}]
		⋮
+		[,SDSASZE={**0K**|number}]
		[,SEC={**YES**|NO~~|MIGRATE~~}]
		⋮
		[,SNSCOPE={**NONE**|CICS|MVSIMAGE|SYSPLEX}]
		⋮
		[,SPCTRxx={(1[,2][,3])|ALL|OFF}]
		[,STNTRxx={(1[,2][,3])|ALL|OFF}]
		⋮
		[,TCSACTN={**NONE**|UNBIND}]
		[,TCSWAIT={**4**|number|NO|NONE|0}]
		[,TD=({**3**|number1}[,{**3**|number2}])]
		[,TRANISO={**NO**|YES}]
		⋮
		[,TS=([COLD][,{0|**3**|value-1}][,{**3**|value-2}])]
		⋮
#		[,UDSASZE={**0M**|number}]
		~~[,USCS={**64K**|number}]~~
		⋮
		[,USRDELAY={**30**|number}]
		⋮
		[,XUSER={**YES**|NO}]
		⋮
	END	DFHSITBA

The primary method of providing system initialization parameters is with a system initialization table (SIT). The parameters of the SIT, which you assemble as a load table, supply the system initialization program with most of the information necessary to initialize the system to suit your unique environment. You can generate more than one SIT, and at the time of system initialization select the one that is appropriate to your needs.

When you start up a CICS region, you specify which SIT you want, and other system initialization parameters (with a few exceptions), in any of three ways:

1. In the PARM parameter of the EXEC PGM=DFHSIP statement
2. In the SYSIN data set defined in the startup job stream
3. Through the system operator's console.

Unless otherwise stated in the descriptions of the system initialization parameters, you can use these methods to override the system initialization parameters in the SIT.

+ **APARs PN70228 and PN88030**

+ Documentation for PN70228 added on 27 September 1996. Documentation
+ for PN88030 added on 20 May 1997.

+ **CDSASZE=}0K|number}**

Code this parameter with the size of the CICS dynamic storage area (CDSA).
The default size is 0, indicating that CICS can change the DSA size
dynamically. A non-ero value indicates that the DSA size is fixed.

+ **number**

+ Specify number as an amount of storage in the range 0 to 16 777 215
+ bytes in multiples of 262 144 bytes (256KB). If the size specified is not a
+ multiple of 256KB, CICS rounds the value up to the next multiple.

+ You can specify number in bytes (for example, 4 194 304), as a whole
+ number of kilobytes (for example, 4096K), or as a whole number of
+ megabytes (for example, 4M).

CMDPROT={YES|NO}

Code this parameter to allow, or inhibit, CICS validation of start addresses of storage referenced as output parameters on EXEC CICS commands.

YES

If you specify YES, CICS validates the initial byte at the start of any storage that is referenced as an output parameter on EXEC CICS commands to ensure that the application program has write access to the storage. This ensures that CICS does not overwrite storage on behalf of the application program when the program itself cannot do so. If CICS detects that an application program has asked CICS to write into an area to which the application does not have addressability, CICS abends the task with an AEYD abend.

The level of protection against bad addresses depends on the level of storage protection in the CICS environment. The various levels of protection provided when you specify CMDPROT=YES are shown in Table 3 on page 31.

NO

If you specify NO, CICS does not perform any validation of addresses of the storage referenced by EXEC CICS commands. This means that an application program could cause CICS to overwrite storage to which the application program itself does not have write access.

CONFDATA={SHOW|HIDETC}

Code this parameter to indicate whether CICS is to suppress (hide) user data that might otherwise appear in CICS trace entries or in dumps that contain the RAIA. This option applies to initial input data received on a VTAM RECEIVE ANY operation, the initial input data received on an MRO link, and FEPI screens and RPLAREAs.

SHOW

Data suppression is not in effect. User data is traced regardless of the CONFDATA option specified in transaction resource definitions. This option overrides the CONFDATA option in transaction resource definitions.

HIDETC

This specifies that you want CICS to 'hide' user data from CICS trace entries. It also indicates that VTAM RAIAs are to be suppressed from CICS dumps. The action actually taken by CICS is subject to the individual CONFDATA attribute on the transaction resource definition (see Table 58 on page 631).

If you specify CONFDATA=HIDETC, CICS processes VTAM, MRO, and FEPI user data as follows:

- **VTAM**: CICS clears the VTAM RAIA containing initial input as soon as it has been processed, and before the target transaction has been identified.

  The normal trace entries (FC90 and FC91) are created on completion of the RECEIVE ANY operation with the text "SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT" replacing all the user data except the first 4 bytes of normal data, or the first 8 bytes of function management headers (FMHs).

  CICS then identifies the target transaction for the data. If the transaction definition specifies CONFDATA(NO), CICS traces the user data that it suppressed from the FC90 trace in the trace entry AP FC92. This trace entry is not created if the transaction is defined with CONFDATA(YES).

- **MRO**: CICS does not trace the initial input received on an MRO link.

  The normal trace entries (DD16, DD23, and DD25) are created with the text "SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT" replacing all the user data.

  CICS then identifies the target transaction for the data. If the transaction definition specifies CONFDATA(NO), CICS traces the user data that it suppressed from DD16 in the trace entry AP FC92. This special trace entry is not created if the transaction is defined with CONFDATA(YES).

- **FEPI**: FEPI screens and RPL data areas (RPLAREAs) areas are suppressed from all FEPI trace points if CONFDATA(YES) is specified in the transaction resource definition. The user data in the FEPI trace

points AP 1243, AP 1244, AP 145E, AP 145F, AP 1460, AP 1461, AP 1595, AP 1596, AP 1597, AP 1598, and AP 1599 is replaced with the message "SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT." If the transaction definition specifies CONFDATA(NO), the FEPI trace entries are created with the user data as normal.

Mirror transactions: The CICS-supplied mirror transaction definitions are specified with CONFDATA(YES). This ensures that, when you specify CONFDATA=HIDETC as a system initialization parameter, CICS regions running mirror transactions suppress user data as described for VTAM and MRO data.

Modified data: By waiting until the transaction has been identified to determine the CONFDATA option, VTAM or MRO data may have been modified (for example, it may have been translated to upper case).

The interaction between the CONFDATA system initialization parameter and the CONFDATA attribute on the transaction resource definition is shown in Table 58.

Table 58. Effect of CONFDATA system initialization and transaction definition parameters

CONFDATA on transaction	CONFDATA system initialization parameter	
	SHOW	**HIDETC**
NO	Data not suppressed	VTAM RAIAs are cleared. Initial input of VTAM and MRO data is suppressed from the normal FC90, FC91, DD16, DD23, and DD25 trace entries. For FC90 and DD16 traces only, suppressed user data is traced separately in an FC92 trace entry. FEPI screens and RPLAREAs are traced as normal.
YES	Data not suppressed	VTAM RAIAs are cleared. All VTAM, MRO, and FEPI user data is suppressed from trace entries.

You cannot modify the CONFDATA option while CICS is running. You must restart CICS to make such a change.

Restrictions

You can code the CONFDATA parameter in the SIT, PARM, and SYSIN only.

CONFTXT={NO|YES}

Code this parameter to indicate whether CICS is to prevent VTAM from tracing user data.

NO

CICS does not prevent VTAM from tracing user data.

YES

CICS prevents VTAM from tracing user data.

Restrictions

You can code the CONFTXT parameter in the SIT, PARM, and SYSIN only.

DSALIM={5M|number}

Specifies the upper limit of the total amount of storage within which CICS can allocate the individual dynamic storage areas (DSAs) that reside below the 16MB boundary.

5M

The default DSA limit is 5MB (5 242 880).

number

Specify *number* as an amount of storage in the range 2MB to 16MB (2 097 152 bytes to 16 777 216 bytes) in multiples of 262 144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify *number* in bytes (for example, 4 194 304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

From the storage size that you specify on the DSALIM parameter, CICS allocates the following dynamic storage areas:

The user DSA (UDSA)

The user-key storage area for all user-key task-lifetime storage below the 16MB boundary.

The read-only DSA (RDSA)

The key-0 storage area for all reentrant programs and tables below the 16MB boundary.

The shared DSA (SDSA)

The user-key storage area for any non-reentrant user-key RMODE(24) programs, and also for any storage obtained by programs issuing EXEC CICS GETMAIN commands for storage below the 16MB boundary with the SHARED option.

The CICS DSA (CDSA)

The CICS-key storage area for all non-reentrant CICS-key RMODE(24) programs, all CICS-key task-lifetime storage below the 16MB boundary, and for CICS control blocks that reside below the 16MB boundary.

Notes:

1. CICS allocates the UDSA in multiples of 1MB when transaction isolation is active, but in multiples of 256KB in CICS regions without transaction isolation. The other DSAs below 16MB are allocated in multiples of 256KB, with or without transaction isolation. The maximum you can specify depends on a number of factors, such as how you have configured your MVS storage (which governs how much private storage remains below the line) and how much private storage you must leave free to satisfy MVS GETMAIN requests for storage outside the DSAs.
2. For information about calculating the amount of storage to specify on the DSALIM parameter, see the *CICS/ESA Performance Guide*.

DSHIPIDL={020000|hhmmss}

Specifies the minimum time, in hours, minutes, and seconds, that an *inactive* shipped terminal definition must remain installed in this region. When the timeout delete transaction is invoked, only those shipped definitions that have been inactive for longer than the specified time are deleted.

You can use this parameter in a transaction routing environment, on the application-owning and intermediate regions, to prevent terminal definitions having to be reshipped because they have been deleted prematurely.

hhmmss Specify a 1 to 6 digit number in the range 0–995959. Numbers that have fewer than six digits are padded with leading zeros.

DSHIPINT={120000|0|hhmmss}

Specifies the interval between invocations of the timeout delete transaction. The timeout delete transaction removes any shipped terminal definitions that have not been used for longer than the time specified by the DSHIPIDL parameter.

You can use this parameter in a transaction routing environment, on the application-owning and intermediate regions, to control:

- How often the timeout delete transaction is run.
- The approximate time of day at which a mass delete operation is to take place, relative to CICS startup.

  Note: For more flexible control over when mass delete operations take place, you can use a CEMT SET DELETSHIPPED or EXEC CICS SET DELETSHIPPED command to reset the interval. (The revised interval starts *from the time the command is issued*, **not** from the time the remote delete transaction was last invoked, nor from CICS startup.)

0 The timeout delete transaction is not invoked. You might set this value in a terminal-owning region, or if you are not using shipped definitions.

hhmmss Specify a 1 to 6 digit number in the range 1–995959. Numbers that have fewer than six digits are padded with leading zeros.

DTRTRAN={CRTX|name}

This is the name of the transaction definition that you want CICS to use for dynamic transaction routing. This is intended primarily for use in a CICS terminal-owning region, although you can also use it in an application-owning region when you want to daisy-chain transaction routing requests. In a dynamic transaction routing environment, the transaction named on DTRTRAN must be installed in the CICS terminal-owning regions if you want to eliminate the need for resource definitions for individual transactions.

See the description of the new CRTX transaction on page 496 for information about defining a dynamic transaction routing definition.

The transaction name is stored in the catalog for recovery during CICS restarts.

CRTX

This is the default dynamic transaction definition. It is the name of the CICS-supplied sample transaction resource definition provided in the CSD group DFHISC.

name
The name of your own dynamic transaction resource definition that you want CICS to use for dynamic transaction routing.

ECDSASZE=}0K|number}
Code this parameter with the size of the extended CICS dynamic storage area (ECDSA). The default size is 0, indicating that CICS can change the DSA size dynamically. A non-ero value indicates that the DSA size is fixed.

number
Specify number as an amount of storage in the range 0 to 1 073 741 824 bytes in multiples of 1 048 576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4 194 304), as a whole number of kilobytes (for example, 4096K), or as a whole number of megabytes (for example, 4M).

EDSALIM={20M|number}
Specifies the upper limit of the total amount of storage within which CICS can allocate the individual extended dynamic storage areas (EDSAs) that reside above the 16MB boundary.

20M
The default EDSA limit is 20MB (20 971 520 bytes).

number
Specify *number* as a value in the range 10MB to 2047MB, in multiples of 1MB. If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify *number* in bytes (for example, 33 554 432), or as a whole number of kilobytes (for example, 32 768K), or a whole number of megabytes (for example, 32M).

The maximum value allowed depends on a number of factors, such as:

- The size of the region you have specified on the MVS REGION parameter in the CICS job or procedure.
- How much storage you require for the CICS internal trace table.
- How much private storage you must leave free to satisfy MVS GETMAIN requests for storage above the 16MB boundary outside the DSAs.

From the storage value that you specify on the EDSALIM parameter, CICS allocates the following extended dynamic storage areas:

The extended user DSA (EUDSA)
The user-key storage area for all user-key task-lifetime storage above the 16MB boundary.

The extended read-only DSA (ERDSA)
The key-0 storage area for all reentrant programs and tables above the 16MB boundary.

The extended shared DSA (ESDSA)
The user-key storage area for any non-reentrant user-key RMODE(ANY) programs, and also for any storage obtained by

programs issuing CICS GETMAIN commands for storage above the 16MB boundary with the SHARED option.

The extended CICS DSA (ECDSA).
The CICS-key storage area for all non-reentrant CICS-key RMODE(ANY) programs, all CICS-key task-lifetime storage above the 16MB boundary, and CICS control blocks that reside above the 16MB boundary.

CICS allocates all the DSAs above the 16MB boundary in multiples of 1MB.

Note: For information about calculating the amount of storage to specify on the EDSALIM parameter, see the *CICS/ESA Performance Guide*. For example, the value that you specify must allow for all temporary storage MAIN requests, for which CICS uses the ECDSA.

ERDSASZE=}0K|number}
Code this parameter with the size of the extended read-only dynamic storage area (ERDSA). The default size is 0, indicating that CICS can change the DSA size dynamically. A non-ero value indicates that the DSA size is fixed.

number
Specify number as an amount of storage in the range 0 to 1 073 741 824 bytes in multiples of 1 048 576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4 194 304), as a whole number of kilobytes (for example, 4096K), or as a whole number of megabytes (for example, 4M).

ESDSASZE=}0K|number}
Code this parameter with the size of the extended shared dynamic storage area (ESDSA). The default size is 0, indicating that CICS can change the DSA size dynamically. A non-ero value indicates that the DSA size is fixed.

number
Specify number as an amount of storage in the range 0 to 1 073 741 824 bytes in multiples of 1 048 576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4 194 304), as a whole number of kilobytes (for example, 4096K), or as a whole number of megabytes (for example, 4M).

EUDSASZE=}0K|number}
Code this parameter with the size of the extended user dynamic storage area (EUDSA). The default size is 0, indicating that CICS can change the DSA size dynamically. A non-ero value indicates that the DSA size is fixed.

number
Specify number as an amount of storage in the range 0 to 1 073 741 824 bytes in multiples of 1 048 576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4 194 304), as a whole number of kilobytes (for example, 4096K), or as a whole number of megabytes (for example, 4M).

GNTRAN={CESF|transaction_id}

Specifies the transaction that you want CICS to invoke when a user's terminal-timeout period expires.

CESF

The default, CESF, is the basic CICS signoff transaction without any options. This transaction attempts to sign off a terminal, subject to the SIGNOFF attribute of the TYPETERM resource definition for that terminal.

transaction_id

The name of an alternative timeout transaction to signoff the user at the timed-out terminal. Specifying your own transaction allows you to specify functions in addition to, or instead of, signoff. For example, your own transaction could issue a prompt for the terminal user's password, and allow the session to continue if the correct password is entered.

Notes:

1. When either the CESF transaction or your own, specified, transaction attempts to sign off a terminal, the result is subject to the SIGNOFF attribute of the TYPETERM resource definition for the terminal, as follows:

SIGNOFF	Effect
YES	The terminal is signed off, but not logged off.
NO	The terminal remains signed on and logged on.
LOGOFF	The terminal is both signed off and logged off.

2. If you use security on your CICS region, you should not specify CESN on the GNTRAN system initialization parameter because the CESN transaction does not sign off the signed-on user until a valid or an invalid attempt to sign on again is made.

GRNAME=name

Specifies the VTAM generic resource name under which a group of CICS terminal-owning regions in a CICSplex register to VTAM.

There is no default for GRNAME. If you don't specify GRNAME, CICS does not register itself with the VTAM generic resources function.

Notes:

1. If you are operating a CICSplex that comprises separate terminal-owning regions and application-owning regions, you should ensure that you define a VTAM generic resource name to the CICS terminal-owning regions only.
2. The generic resource name can be 1 through 8 characters. However, when defining a generic resource name to VTAM you are recommended to pad the name to the full 8 characters with a generic symbol, as shown in the following example:

```
*********************************************************************
*    Generic resources APPL definition for CICS Terminal-Owning
*                 regions in the Dallas  CICSplex
*********************************************************************
CICSD###  APPL  AUTH=(ACQ,VPACE,PASS),...
```

3. The name you specify to CICS must match the generic resource name defined to VTAM. For example, for the APPL definition shown above:

```
GRNAME=CICSD###
```

4. If you specify the XRF=YES parameter, you should not specify a value for the GRNAME system initialization parameter. Any value specified for GRNAME is set to blanks.
5. If you specify the XRF=NO parameter, and a value for GRNAME, you should not specify a specific applid for the APPLID system initialization parameter. Any specific applid specified for the APPLID parameter is set to the generic applid.

The examples used here are based on a CICS naming convention described in the *System/390 MVS Sysplex Application Migration* manual, GC28-1211.

GRPLIST={DFHLIST|name|(name[,name2][,name3][,name4])}

Specifies the names (each 1 through 8 characters) of up to four lists of resource definition groups on the CICS system definition (CSD) file. The resource definitions in all the groups in the specified lists are loaded during initialization when CICS performs a cold start. If a warm or emergency start is performed, the resource definitions are derived from the global catalog, and the GRPLIST parameter is ignored.

Each name can be either a real group list name or a generic group list name that incorporates global filename characters (+ and *). If you specify more than one group list (either by specifically coding two or more group list names or by coding a group list name with global filename characters), the later group lists are concatenated onto the first group list. Any duplicate system initialization parameters in later group lists override those in earlier group lists.

Use the CEDA command LOCK to protect the lists of resource groups specified on the GRPLIST parameter.

The default is DFHLIST, the CICS-supplied list that specifies the set of resource definitions needed by CICS. If you create your own group list, either add to it the groups specified in DFHLIST (omitting only those for CICS functions that you know you do not need) or specify the DFHLIST name on the GRPLIST parameter. Do not code GRPLIST=NO unless you have a group list named NO.

For more information about the GRPLIST parameter, see the *CICS/ESA System Definition Guide*.

ICVR={5000|number}

specifies the default runaway task time interval in milliseconds as a decimal number. You can code zero, or a number in the range 500 through 2 700 000, in multiples of 500. CICS rounds down values that are not multiples of 500. This is the RUNAWAY interval used by transactions defined with RUNAWAY=SYSTEM (see the *CICS/ESA Resource Definition Guide*). CICS may purge a task if it has not given up control after the RUNAWAY interval for the transaction (or ICVR if the transaction definition specified RUNAWAY=SYSTEM). If you code ICVR=0, runaway task control is inoperative for transactions specifying RUNAWAY=SYSTEM in their transaction definition (that is, tasks do not get purged if they appear to be looping). The ICVR value is independent of the ICV value, and can be less than the ICV value. Note that CICS runaway task detection is based upon task time, that is, the interval is decremented only when the task has control of the processor. For information about commands that reinitialize the ICVR value, see the *CICS/ESA Problem Determination Guide*.

LLACOPY={YES|NO|NEWCOPY}

Code this to indicate whether CICS is to use the LLACOPY macro or the BLDL macro when locating modules in the DFHRPL concatenation.

YES

CICS always uses the LLACOPY macro when locating modules in the DFHRPL concatenation.

NO

CICS always uses the BLDL macro when locating modules in the DFHRPL concatenation.

NEWCOPY

CICS uses the LLACOPY only when a NEWCOPY or a PHASEIN is being performed. At all other times, CICS uses the BLDL macro when locating modules in the DFHRPL concatenation.

Notes:

1. If you code LLACOPY=NO or LLACOPY=NEWCOPY you can still benefit from having LLA managed data sets within your DFHRPL concatenation. Modules will continue to be loaded from VLF if appropriate.
2. If an LLA managed module has been altered, a BLDL macro may not return the new information and a subsequent load will still return the old copy of the module. To load the new module, an LLACOPY must be issued against that module or a MODIFY LLA,REFRESH command must be issued on a system console.

MNCONV={NO|YES}

Specifies whether or not conversational tasks are to have separate performance class records produced for each pair of terminal control I/O requests.

Any clock (including user-defined) that is active at the time such a performance class record is produced is stopped immediately before the record is written. After the record is written, such a clock is reset to zero and restarted. Thus a clock whose activity spans more than one recording interval within the conversational task appears in multiple records, each showing part of the time, and the parts adding up to the total time the clock is active. The high-water-mark fields (which record maximum levels of storage used) are reset to their current values. All other fields are set to X'00', except for the key fields (transid, termid). The monitoring converse status is recorded in the CICS global catalog for use during warm and emergency restarts.

MNFREQ={0|hhmmss}

Specifies the interval for which CICS automatically produces a transaction performance class record for any long-running transaction. The monitoring frequency value is recorded in the CICS global catalog for use during warm and emergency restarts.

0 means that no frequency monitoring is active.

hhmmss

is the interval for which monitoring produces automatically a transaction performance class record for any long-running transaction. Specify a 1 to 6 digit number in the range 001500–240000. Numbers that are fewer than six digits are padded with leading zeroes.

MNSUBSYS={null|xxxx}
Specifies the 4-character name to be used as the subsystem identification in the monitoring SYSEVENT class records. If you do not specify a name, the subsystem identification defaults to the first four characters of the CICS generic APPLID. The monitoring subsystem id is recorded in the CICS global catalog for use during warm and emergency restarts.

For more information on the SYSEVENT class of monitoring data and the subsystem identification, and about the implications for SYSEVENT recording in a MVS Workload Manager environment, see the *CICS/ESA Performance Guide*.

MNSYNC={NO|YES}
Specifies whether or not you want CICS to produce a transaction performance class record when a transaction takes an implicit or explicit syncpoint (unit-of-work). No action is taken for syncpoint rollbacks. The monitoring syncpoint status is recorded in the CICS global catalog for use during warm and emergency restarts.

MNTIME={GMT|LOCAL}
Specifies whether you want the time stamp fields in the performance class monitoring data to be returned to an application using the EXEC CICS COLLECT STATISTICS MONITOR(taskno) command in either GMT or local time. The monitoring time value is recorded in the CICS global catalog for use during warm and emergency restarts.

For more information on the EXEC CICS COLLECT STATISTICS command, see the *CICS/ESA System Programming Reference*.

MXT={5|number}
Specifies the maximum number, in the range 1 through 999, of ***user*** tasks CICS allows to exist at any time. CICS queues requests for tasks above this number but does not action (attach) them until the number of tasks attached drops below the MXT limit.

Note: The MXT value does **not** include CICS system tasks.

PGAICTLG={MODIFY|NONE|ALL}
Specifies whether autoinstalled program definitions should be cataloged. While CICS is running, you can set whether autoinstalled programs should be cataloged dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

MODIFY
Autoinstalled program definitions are cataloged only if the program definition is modified by a SET PROGRAM command subsequent to the autoinstall.

NONE
Autoinstalled program definitions are not cataloged. This gives a faster CICS restart (warm and emergency) compared with the MODIFY or ALL options, because CICS does not reinstall definitions from the global catalog. Definitions are autoinstalled on first reference.

ALL
Autoinstalled program definitions are written to the global catalog at the time of the autoinstall, and following any subsequent modification.

PGAIEXIT={DFHPGADX|name}
Specifies the name of the program autoinstall exit program. While CICS is running, you can set the name of the program autoinstall exit program dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

PGAIPGM={INACTIVE|ACTIVE}
Specifies the state of the program autoinstall function at initialization. While CICS is running, you can set the status of program autoinstall dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

INACTIVE
The program autoinstall function is disabled.

ACTIVE
The program autoinstall function is enabled.

PLTPISEC={NONE|CMDSEC|RESSEC|ALL}
Specifies whether or not you want CICS to perform command security or resource security checking for PLT programs during CICS initialization. The PLT programs run under the authority of the userid specified on PLTPIUSR, which must be authorized to the appropriate resources defined by PLTPISEC.

NONE
Specifies that you do not want any security checking on on PLT initialization programs.

CMDSEC
Specifies that you want CICS to perform command security checking only.

RESSEC
Specifies that you want CICS to perform resource security checking only.

ALL
Specifies that you want CICS to perform both command and resource security checking.

Restrictions

You can code the PLTPISEC parameter in the SIT, PARM, or SYSIN only.

PLTPISEC={NONE|CMDSEC|RESSEC|ALL}
Specifies whether or not you want CICS to perform command security or resource security checking for PLT programs during CICS initialization. The PLT programs run under the authority of the userid specified on PLTPIUSR, which must be authorized to the appropriate resources defined by PLTPISEC.

NONE
specifies that you do not want any security checking on on PLT initialization programs.

CMDSEC
specifies that you want CICS to perform command security checking only.

RESSEC
specifies that you want CICS to perform resource security checking only.

ALL
specifies that you want CICS to perform both command and resource security checking.

Restrictions

You can code the PLTPISEC parameter in the SIT, PARM, or SYSIN only.

PLTPIUSR=userid
Specifies the userid that CICS is to use for security checking for PLT programs that run during CICS initialization. All PLT programs run under the authority of the specified userid, which must be authorized to all the resources referenced by the programs, as defined by the PLTPISEC parameter.

PLT programs are run under the CICS internal transaction, CPLT. Before the CPLT transaction is attached, CICS performs a surrogate user check against the CICS region userid (the userid under which the CICS region is executing). This is to ensure that the CICS region is authorized as a surrogate of the userid specified on the PLTPIUSR parameter. This ensures that you cannot arbitrarily specify any PLT userid in any CICS region—each PLT userid must first be authorized to the appropriate CICS region.

If you do not specify the PLTPIUSR parameter, CICS runs PLTPI programs under the authority of the CICS region userid, in which case CICS does not perform a surrogate user check. However, the CICS region userid must be authorized to all the resources referenced by the PLT programs.

Restrictions

You can code the PLTPIUSR parameter in the SIT, PARM, or SYSIN only.

PSDINT={0|hhmmss}
Specifies the persistent session delay interval. This delay interval specifies if, and for how long, VTAM is to hold sessions in a recovery-pending state if CICS fails. The value for hours can be in the range 0 through 23; the minutes and seconds in the range 00 through 59 inclusive.

This value can be overridden during CICS execution (and hence change the action taken by VTAM if CICS fails).

0 A zero value specifies that, if CICS fails, sessions are terminated. This is the default.

hhmmss
Specifies a persistent session delay interval from 1 second up to the maximum of 23 hours 59 minutes and 59 seconds. If CICS fails, VTAM holds sessions in recovery pending state for up to the interval specified on the PSDINT system initialization parameter.

Specify a 1-to-6 digit time in hours, minutes and seconds, up to the maximum time. If you specify less than six digits, CICS pads the value with leading zeros. Thus a value of 500 is taken as five minutes exactly.

The interval you specify must cover the time from when CICS fails to when the VTAM ACB is opened by CICS during the subsequent emergency restart.

VTAM holds all sessions in recovery pending state for up to the interval specified (unless they are unbound through path failure or VTAM operator

action, or other-system action in the case of intelligent LUs). The PSDINT value used must take account of the types and numbers of sessions involved.

You must exercise care when specifying large PSDINT values because of the problems they may give in some environments, in particular:

- Dial-up sessions—real costs may be incurred
- LU6.2 sessions to other host systems—such systems may become stressed.

Notes:

1. When specifying a PSDINT value, you must consider the number and, more particularly, the nature of the sessions involved. If LU6.2 sessions to other host systems are retained in recovery pending state, the other host systems may experience excessive queuing delays. This point applies to LU6.1 sessions which are retained until restart (when they are unbound).
2. The PSDINT parameter is incompatible with the XRF=YES parameter. If XRF=YES is specified, the PSDINT parameter is ignored.

+ **APAR PN70228**

+ Documentation for PN70228 added on 27 September 1996

+ **RDSASZE=}0K|number}**

Code this parameter with the size of the read-only dynamic storage area
(RDSA). The default size is 0, indicating that CICS can change the DSA size
dynamically. A non-zero value indicates that the DSA size is fixed.

+ **number**

+ Specify number as an amount of storage in the range 0 to 16 777 215
+ bytes in multiples of 262 144 bytes (256KB). If the size specified is not a
+ multiple of 256KB, CICS rounds the value up to the next multiple.

+ You can specify number in bytes (for example, 4 194 304), as a whole
+ number of kilobytes (for example, 4096K), or as a whole number of
+ megabytes (for example, 4M).

+ **SDSASZE=}0K|number}**

Code this parameter with the size of the shared dynamic storage area (SDSA).
The default size is 0, indicating that CICS can change the DSA size
dynamically. A non-zero value indicates that the DSA size is fixed.

+ **number**

+ Specify number as an amount of storage in the range 0 to 16 777 215
+ bytes in multiples of 262 144 bytes (256KB). If the size specified is not a
+ multiple of 256KB, CICS rounds the value up to the next multiple.

+ You can specify number in bytes (for example, 4 194 304), as a whole
+ number of kilobytes (for example, 4096K), or as a whole number of
+ megabytes (for example, 4M).

SEC={YES|NO}

In CICS/ESA 4.1, the MIGRATE option has been removed.

Code this parameter to indicate what level of external security you want CICS to use. For more information about this parameter, see the *CICS/ESA System Definition Guide*.

SNSCOPE={NONE|CICS|MVSIMAGE|SYSPLEX}

Specifies whether or not a userid can be signed on to CICS more than once, within the scope of:

- A single CICS region
- A single MVS image
- A sysplex.

NONE

Each userid can be used to sign on for any number of sessions on any CICS region. This is the compatibility option, providing the same signon scope as in releases of CICS before CICS/ESA 4.1.

CICS

Each userid can be signed on once only in the same CICS region. A signon request is rejected if the userid is already signed on to the same CICS region. However, the userid can be used to signon to another CICS region in the same, or another, MVS image.

MVSIMAGE

Each userid can be signed on once only, and to only one of the set of CICS regions in the same MVS image that also specify SNSCOPE=MVSIMAGE. A signon request is rejected if the user is already signed on to another CICS region in the same MVS image.

SYSPLEX

Each userid can be signed on once only, and to only one of the set of CICS regions within an MVS sysplex that also specify SNSCOPE=SYSPLEX. A signon is rejected if the user is already signed on to another CICS region in the same MVS sysplex.

The signon scope (if specified) applies to all userids signing on by an explicit signon request (for example, by an EXEC CICS SIGNON command or the CESN transaction). SNSCOPE is restricted to users signing on at local terminals, or signing on after using the CRTE transaction to connect to another system.

Signon scope specified by SNSCOPE *does not* apply to:

- Non-terminal users.
- The CICS default userid, specified by the DFLTUSER system initialization parameter.
- Preset userids, specified in the USERID option of the DEFINE TERMINAL command.
- Userids for remote users, received in attach headers.
- Userids for link security. For information about which userid is used for link security on a specific connection, see the *CICS/ESA CICS-RACF Security Guide*.
- The userid specified on the PLTPIUSR system initialization parameter.
- The CICS region userid.

Restrictions

You can code the SNSCOPE parameter in the SIT, PARM, or SYSIN only.

SPCTRxx={(1,2|1[,2][,3])|ALL|OFF}

Code this parameter to set the level of tracing for a particular CICS component used by a transaction, terminal, or both, selected for special tracing. You identify the component by coding a value for xx in the keyword. You code one SPCTRxx keyword for each component you want to define selectively. For a CICS component being specially traced that does not have its trace level set by SPCTRxx, the trace level is that set by SPCTR (which, in turn, defaults to (1,2)). You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels. The CICS component codes that you can code for xx on the SPCTRxx keyword are shown in Table 59:

Table 59. CICS component names and abbreviations

Code	Component name	Code	Component name
AP	Application domain	BM	Basic mapping support
BF	Built-in function	CP	Common programming interface
DC	Dump compatibility layer	DD	Directory manager domain
DI	Batch data interchange	DM	Domain manager domain
DS	Dispatcher domain	DU	Dump domain
EI	Exec interface	FC	File control
GC	Global catalog domain	IC	Interval control
IS	Inter-system communication	JC	Journal control
KC	Task control	KE	Kernel
LC	Local catalog domain	LD	Loader domain
LM	Lock domain	ME	Message domain
MN	Monitoring domain	PA	Parameter domain
PC	Program control	PG	Program manager domain
SC	Storage control	SM	Storage domain
SP	Sync point	ST	Statistics domain
SZ	Front end programming interface	TC	Terminal control
TD	Transient data	TI	Timer domain
TR	Trace domain	TS	Temporary storage
UE	User exit interface	US	User domain
XM	Transaction manager domain	XS	Security manager domain

Note: The component codes BF, BM, CP, DC, DI, EI, FC, IC, IS, JC, KC, PC, SC, SP, TC, TD, TS, and UE are sub-components of the AP domain. As such, the corresponding trace entries will be produced with a point ID of AP nnnn.

For more information, see the *CICS/ESA Problem Determination Guide*.

number Code the level numbers for the level of special tracing you want for the CICS component indicated by xx. The options are: 1, (1,2), or (1,2,3).

ALL Code ALL to indicate that you want all the available levels of special CICS tracing switched on for the specified component.

OFF Code OFF to switch off all levels of special CICS tracing for the CICS component indicated by xx.

Restrictions

You can code the SPCTRxx parameter in PARM, SYSIN, or CONSOLE only.

STNTRxx={(1,21[,2][,3])|ALL|OFF}

Specifies the level of standard tracing you require for a particular CICS component. You identify the component by coding a value for xx in the keyword. You code one STNTRxx keyword for each component you want to define selectively. For a CICS component being specially traced that does not have its trace level set by STNTRxx, the trace level is that set by STNTR (which, in turn, defaults to 1). You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels.

The CICS component codes that you can code for xx on this STNTRxx keyword are shown in Table 59 on page 644.

number Code the level number(s) for the level of standard tracing you want for the CICS component indicated by xx. The options are: 1, (1,2), or (1,2,3).

ALL Code ALL to indicate that you want all the available levels of standard tracing switched on for the specified component.

OFF Code OFF to switch off all levels of standard CICS tracing for the CICS component indicated by xx.

Warning: If you select tracing levels 3 or ALL for the storage manager (SM) component, the performance of your CICS system will be degraded. This is because options 3 and ALL switch on levels of trace that are also used for field engineering purposes. See the *CICS/ESA Problem Determination Guide* for further information about the effects of trace levels 3 and ALL.

Restrictions

You can code the STNTRxx parameter in PARM, SYSIN, or CONSOLE only.

TCSACTN={NONE|UNBIND}

Code this parameter with the required action that CICS terminal control should take if the terminal control shutdown wait threshold expires. For details of the wait threshold, see the TCSWAIT system initialization parameter. TCSACTN only takes effect when TCSWAIT is coded with a value in the range 1 through 99. This parameter only applies to VTAM terminals (including LU Type 6.2 single-session APPC terminals), not VTAM intersystem connections (LU Type 6.1 and LU Type 6.2 parallel connections). This is a global default action. On a terminal-by-terminal basis, you can code a DFHZNEP routine to override this action.

NONE

No action is taken. This can be overridden by DFHZNEP.

UNBIND
CICS Terminal Control attempts to force-close the session by issuing a VTAM CLSDST and sending an SNA UNBIND command to the hung terminal. This can be overridden by DFHZNEP.

TCSWAIT={4|number|NO|NONE|0}
Code this parameter with the required CICS terminal control shutdown wait threshold. The wait threshold is the time, during shutdown, that CICS terminal control allows to pass before it considers terminal shutdown to be hung. If all VTAM sessions shutdown and close before the threshold expires then the CICS shutdown process moves on to its next stage, and the terminal control wait threshold then no longer applies. If, however, some of the VTAM session do not complete shutdown and close, then CICS takes special action with these sessions. For details of this special action see the description of the TCSACTN system initialization parameter. The wait threshold only applies to VTAM sessions; that is, VTAM terminals and VTAM intersystem connections. The wait time is specified as a number of minutes, in the range 1 through 99. As a special case, TCSWAIT=NO may be specified to indicate that terminal control shutdown is never to be considered hung, no matter how long the shutdown and close process takes. TCSWAIT=NONE and TCSWAIT=0 are alternative synonyms for TCSWAIT=NO, and all three have the same effect (internally they are held as the one value 0 (zero)).

TD=({3|decimal-value-1}[,{3|decimal-value-2}])
Specifies the number of VSAM buffers and strings to be used for intrapartition transient data (TD).

decimal-value-1
The number of buffers to be allocated for the use of intrapartition transient data. The value must be in the range 1 through 32 767. The default value is 3.

CICS obtains, above 16MB, storage for the TD buffers in units of the page size (4KB). Because CICS optimizes the use of the storage obtained, TD may allocate more buffers than you specify, depending on the control interval (CI) size you have defined for the intrapartition data set.

For example, if the CI size is 1536, and you specify 3 buffers (the default number), CICS actually allocates 5 buffers. This is because 2 pages (8192 bytes) are required to obtain sufficient storage for three 1536-byte buffers, a total of only 4608 bytes, which would leave 3584 bytes of spare storage in the second page. In this case, CICS allocates another 2 buffers (3072 bytes) to minimize the amount of unused storage. In this way CICS makes use of storage that would otherwise be unavailable for any other purpose.

decimal-value-2
The number of VSAM strings to be allocated for the use of intrapartition transient data. The value must be in the range 1 through 255, and must not exceed the value specified in decimal-value-1. The default value is 3.

For example, TD=(8,5) specifies 8 buffers and 5 strings.

The operands of the TD parameter are positional. You must code commas to indicate missing operands if others follow. For example, TD=(,2) specifies the number of strings and allows the number of buffers to default.

TRANISO={NO|YES}

Code this parameter, together with the STGPROT system initialization parameter, to specify whether you want transaction isolation in the CICS region. The permitted values are NO (the default), or YES:

NO

This is the default. If you specify NO, or allow this parameter to default, CICS operates without transaction isolation, and all storage in the CICS address space is addressable as in earlier releases. If you specify STGPROT=YES and TRANISO=NO, CICS storage protection is active without transaction isolation.

YES

Specify YES for transaction isolation. If you specify TRANISO=YES and STGPROT=YES, and you have the required hardware and software, CICS operates with transaction isolation. This ensures that the user-key task-lifetime storage of transactions defined with the ISOLATE(YES) option is isolated from the user-key programs of other transactions.

If you specify TRANISO=YES, but you do not have the required hardware and software or STGPROT=NO is specified, CICS issues an information message during initialization, and operates without transaction isolation.

STGPROT=NO and TRANISO=YES specified in the system initialization table causes an error during assembly (MNOTE 8).

TS=([COLD][,{0|3|decimal-value-1}][,{3|decimal-value-2}])

Specifies:

- Whether or not you want to cold start temporary storage
- The number of VSAM buffers to be used for auxiliary temporary storage
- The number of VSAM strings to be used for auxiliary temporary storage.

COLD

The type of start for the temporary storage program. If you do not want a cold start, code a comma before the second operand.

0 No buffers are required; that is, only MAIN temporary storage is required.

decimal-value-1

The number of buffers to be allocated for the use of auxiliary temporary storage. The value must be in the range 3 through 32 767.

CICS obtains, above 16MB, storage for the auxiliary temporary storage buffers in units of the page size (4KB). Because CICS optimizes the use of the storage obtained, TS may allocate more buffers than you specify, depending on the control interval (CI) size you have defined for the auxiliary temporary storage data set.

For example, if the CI size is 2048, and you specify 3 buffers (the default number), CICS actually allocates 4 buffers. This is because 2 pages (8192 bytes) are required to obtain sufficient storage for three 2048-byte buffers, a total of 6144 bytes, which would leave 2048 bytes of spare storage in the second page. In this case, CICS allocates another 2048-byte buffer to use all of the 8192 bytes obtained. In this way CICS makes use of storage that would otherwise be unavailable for any other purpose.

decimal-value-2
The number of VSAM strings to be allocated for the use of auxiliary temporary storage. The value must be in the range 1 through 255, and must not exceed the value specified in decimal-value-1. The default value is 3.

APAR PN70228

Documentation for PN70228 added on 27 September 1996

UDSASZE=}0K|number}
Code this parameter with the size of the user dynamic storage area (RDSA). The default size is 0, indicating that CICS can change the DSA size dynamically. A non-zero value indicates that the DSA size is fixed.

number
Specify number as an amount of storage in the range 0 to 16 777 215 bytes in multiples of 262 144 bytes (256KB). If the size specified is not a multiple of 256KB (or 1MB if transaction isolation is active), CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4 194 304), as a whole number of kilobytes (for example, 4096K), or as a whole number of megabytes (for example, 4M).

USRDELAY={30|number}
Specify the maximum time, in the range 0 through 10080 minutes (up to 7 days), that an eligible userid and its associated attributes are to be retained in the user table if the userid is unused. An entry in the user table for a userid that is retained during the delay period can be reused.

The userids eligible for reuse within the USRDELAY period are any that are:

- Received from remote systems
- Specified on SECURITYNAME in CONNECTION definitions
- Specified on USERID in SESSIONS definitions
- Specified on USERID on DFHDCT TYPE=INTRA definitions
- Specified on USERID on START commands.

Within the USRDELAY period, a userid in any one of these categories can be reused in one of the other categories, provided the request for reuse is qualified with the same qualifiers. If a userid is qualified by different group id, APPLID, or terminal id, a retained entry is not reused.

If a userid is unused for more than the USRDELAY limit, it is removed from the system, and the message DFHUS0200 is issued. You can suppress this message in an XMEOUT global user exit program. If you specify USRDELAY=0, all eligible userids are deleted immediately after use, and the message DFHUS0200 is not issued.

When running a remote transaction, a userid remains signed-on to the remote CICS region (after the conversation associated with the first attach request is complete) until the delay specified by USRDELAY has elapsed since the last transaction associated with the attach request for the userid has completed. When this event occurs, the userid is removed from the remote CICS region.

For more information about the use of USRDELAY, see the *CICS/ESA Performance Guide*.

XUSER={YES|NO}

Specifies whether or not CICS is to perform surrogate user checks.

YES

specifies that CICS is to perform surrogate user checking in all those situations that require such checks to be made (for example, on EXEC CICS START commands without an associated terminal). For information about the various circumstances in which CICS performs surrogate user checks, see the *CICS/ESA CICS-RACF Security Guide*.

NO

Specifies that CICS is not to perform any surrogate user checking.

Restrictions

You can code the XUSER parameter in the SIT, PARM, or SYSIN only.

Chapter 39. Prerequisite hardware and software for CICS/ESA 4.1

This chapter gives detailed information about related IBM program products that you need either to use CICS/ESA 4.1, or to exploit the all the function in this release. It covers the following topics:

- Minimum levels of hardware
- MVS/ESA operating system
- Data Facility Product (DFP)
- IMS/ESA
- IBM DATABASE 2 (DB2)
- IBM telecommunications access methods
- IBM external security manager (RACF).

Hardware requirements

If you want to run CICS/ESA 4.1 without exploiting special functions, you need an Enterprise Systems Architecture/370 (ESA/370) or an Enterprise Systems Architecture/390 (ESA/390) processor that meets the requirements of the host operating system, CICS, the access methods, and your application programs.

Hardware requirements for storage protection

For the CICS storage protection facility, you need an ESA/390 processor that provides the subsystem storage protection facility, in addition to the prerequisite software. See the latest IBM hardware information for details of machines that support subsystem storage protection.

For the operating system requirements, see “Summary of software requirements for CICS/ESA 4.1” on page 653.

Hardware requirements for transaction isolation

For the CICS transaction isolation facility, you need an ESA/390 processor that includes the subspace-group facility, in addition to the prerequisite software. See the latest IBM hardware information for details of machines that support subspace-groups.

For the operating system requirements, see “Operating system requirements” on page 654.

Hardware requirements for a sysplex

To use CICS/ESA 4.1 in a sysplex, or any form of multisystem data sharing, you must first create a sysplex. The multiple MVS images that comprise a sysplex can

run in either:

- One CPC[11] (the CPC being an ESA/390-capable processing system) partitioned into one or more logical partitions (LPARs) using the PR/SM facility, or
- One or more CPCs (possibly of different processor models), with each CPC running a single MVS image, or
- A mixture of LPARs and separate CPCs.

You also need the following hardware:

- **Channel-to-channel links, ESCON channels or high-speed coupling facility links**—for XCF signaling.
- **External time reference (ETR) facility**—when the sysplex consists of multiple MVS systems running on two or more CPCs, XCF requires that the CPCs be connected to the same ETR facility. XCF uses the synchronized time stamp that the ETR provides for to provide a common sysplex-wide time reference, and for monitoring and sequencing events within the sysplex.
- DASD controllers with enough paths to dedicate one path to each MVS image in the sysplex.
- For multiple system data sharing with DBCTL, you need a coupling facility.

DASD requirements

To use CICS/ESA 4.1 in a sysplex, or any form of multisystem data sharing, you need DASD controllers with enough paths to dedicate one path to each MVS image in the sysplex.

To use T0 (or concurrent) copy of active files or databases, you need to install 3990 model 3 DASD controllers.

Tape requirements

CICS/ESA 4.1 still supports current tape devices, including 3480 devices (although not recommended), for tape logging. However, it does not support the use of 3480 folow-on devices for tape logging.

Note: Future CICS releases will discontinue support for tape logging.

There are no restrictions on the use of tape devices for extrapartition transient data.

Discontinued device support

There is no support in CICS/ESA Version 4 for devices and controllers accessed using BTAM, GAM, or TCAM (ACB).

(VTAM and TCAM (DCB) is supported.)

Current terminal types, and their modes of connection, are listed in the *CICS/ESA Resource Definition Guide*.

---

[11] CPC. One physical processing system, such as the whole of an ES/9000 9021 Model 820, or one physical partition of such a machine. A physical processing system consists of main storage, and one or more central processing units (CPUs), time-of-day (TOD) clocks, and channels, which are in a single configuration. A CPC also includes channel subsystems, service processors, and expanded storage, where installed.

Summary of software requirements for CICS/ESA 4.1

This section summarizes the software levels that you need for CICS/ESA 4.1.

Minimum software levels required

The minimum levels for software products that you can use with CICS/ESA 4.1 are listed in Table 60. The levels are the same as for CICS/ESA 3.3, except where the new minimum level is marked by an asterisk (*).

Table 60. Minimum levels of software products for CICS/ESA 4.1

Software product	CICS/ESA 3.3	CICS/ESA 4.1
ACF/TCAM(DCB) 5735-RC3	V2	V2
+ ACF/VTAM 5685-085	V3.1.1	V3.4.1*
CICSPD 5695-035	V1.2	N/A*
CICSVR 5695-010	V2.1	V2.1
DB2 5665-DB2	V2.2(SPE)	V2.3*
EPDM 5695-101	V1.1.0	V1.1.0
GDDM/MVS 5665-356	V2.3	V2.3
IMS/VS-DB 5665-332	V2.2	N/A*
IMS/ESA-DM 5665-408	V3.1	V3.1
MVS/DFP 5665-XA3	V3.1	V3.1
MVS/ESA(BCP) MVS/SP-JES2 5685-001 MVS/SP-JES3 5685-002	V3.1.3	V3.1.3
+ OS/390	R1	R1
OS PL/I-R/T 5668-910	V2.3	V2.3
+ RACF 5740-XXH	V1.8.1	V1.9.0* + PTFs
SLR 5665-397	V3.3	V3.3 + PTF*
SMP/E	R5	R1.7+PTFs*

Notes:

1. These ***minimum*** prerequisite software levels do not support some new functions in either CICS/ESA 3.3 or CICS/ESA 4.1. For the software levels required for extra functions, see “Software levels required for extra function.”
2. For RACF 1.9.0, apply PTFs OY42716 and OY47909 to correct errors in the mapping of the CICS segment.
3. For SLR V3.3, apply a PTF for APAR PN48839.
4. For SMP/E 1.7, apply the following PTFs:

   UR40251 for FMID HMP1701
   UR40252 for FMID JMP1701
   UR40255 for FMID JMP1711

Software levels required for extra function

The levels for software products that you can use with CICS/ESA 4.1 to exploit extra functions are listed in Table 61 on page 654.

Table 61. Software levels required for extra function

New function description	Introduced in CICS	Software level required
VSAM BWO - SMS only	V3.2.1	MVS/DFP V3.2 DFHSM V2.5, DFDSS V2.5, and CICSVR V2 (+ Info APAR II04910)
XCF with XRF	V3.2.1	MVS/ESA SP V4and MVS/DFP V3.2 or later
CICS TSO console support	V3.2.1	MVS/ESA SP V4
FFDC to SYS1.LOGREC	V3.2.1	MVS/ESA SP V4
Subsystem Storage Protect	V3.3	MVS/ESA SP V4.2.2
Distributed program link	V3.3	PTFs-some servers
IMS-DBCTL with SSP	V3.3	IMS/ESA-DM V4.1
Transaction isolation	V4.1	MVS/ESA 5.1
Coupling communication	V4.1	MVS/ESA 5.1
Workload manager	V4.1	MVS/ESA 5.1
DBCTL n-way data sharing	V4.1	IMS/ESA-DM V5.1
Single-node persistent sessions	V4.1	ACF/VTAM V3.4.1 with APAR OY65251
+ Multi-node persistent sessions	V4.1	ACF/VTAM V4.4
Generic resources	V4.1	ACF/VTAM V4.2 and MVS/ESA 5.1 (also see note below).
Security: performance and FEPI request PassTicket	V4.1	RACF V2.1

| **Note:** To use VTAM generic resources in CICS/ESA 4.1, ACF/VTAM V4.2 (or later) must be:

- Running under an MVS that is part of a sysplex
- Connected to the sypslex coupling facility.

For information about the sysplex coupling facility, see the *MVS/ESA Setting Up a Sysplex* manual, GC28-1449.

Operating system requirements

+ CICS/ESA 4.1 operates under OS/390 or the MVS/ESA operating system.

The level of the MVS operating system that you require depends on the level of CICS function you want to use. The levels of MVS needed to run CICS/ESA 4.1 without exploiting special functions are listed in Table 60 on page 653. The levels of MVS needed to exploit special functions are listed in “Summary of software requirements for CICS/ESA 4.1” on page 653.

Programming requirements

The minimum programming requirements for CICS/ESA 4.1 are listed in Table 60 on page 653. The programming requirements to use special functions of CICS/ESA 4.1 are listed in Table 61.

Compilers and assembler

CICS/ESA 4.1 supports the following assembler, COBOL, PL/I and C/370 compilers:

- MVS Assembler H Version 2 (5668-962)
- VS COBOL II (5668-958 and 5688-023)

  (Requires PTF for APAR PN43097—see the *Program Directory* or the *CICS/ESA Migration Guide* for details of the PTFs
- OS PL/I Optimizing Compiler Version 1 Release 5.1 (5734-PL1)
- OS PL/I Optimizing Compiler Version 2 Release 1 (5668-910) or later
- C/370 (5688-040).

CICS/ESA 4.1 also supports IBM SAA AD/Cycle Language Environment/370 Version 1 Releases 1 and 2 run-time environment (5688-198), with the following COBOL, C/370, and PL/I SAA AD/Cycle compilers:

- SAA AD/Cycle COBOL/370 (5688-197)
- SAA AD/Cycle C/370 (5688-216)
- SAA AD/Cycle PL/I (5688-235)

CICS support for unsupported COBOL compilers

CICS/ESA 4.1 retains translation and execution-time support for application programs compiled by the following unsupported COBOL compilers:

- Full American National Standard Version 4 (5734-CB2)
- OS/VS COBOL (5740-CB1)

Execution-time support is withdrawn for application programs compiled by the old OS/VS COBOL compilers 360S-CB-545 and 5734-CB1.

PL/I execution-time support

For PL/I run-time support of a CICS application program compiled with any PL/I compiler, CICS/ESA 4.1 requires the run-time library from OS PL/I Version 2 Release 3 (5668-910 or 5668-911) or later.

Alternatively, you can use run-time libraries supplied with Language Environment/370 to support run-time execution of programs written in PL/I. The minimum level of Language Environment is Version 1 Release 2. However, to run PL/I programs under Language Environment/370, programs must be link-edited with the linkage editor control statement `REPLACE PLISTART` included.

This applies both to new programs, and to old programs which must be re-linkedited in order to remove PLISTART.

Programming language functions

For the following programming language functions in CICS/ESA 4.1 you need the specified language compilers:

- To use the CICS support for C/370, either of the following compiler and library combinations are required for C language translations:
  - C/370 Compiler (5688-040) and C/370 Library (5668-039) Version 1 Release 2, with APAR PL55900 (PTF UL90404). You must also have

applied the C/370-enabling SPE, which includes PTFs UL90404 and UL64595; see APAR PL55900 for information about applying these PTFs.

  - C/370 Compiler (5688-187) and C/370 Library (5668-188) Version 2 Release 1 or later.

- To use CICS support for COBOL ANSI 85 standards, use VS COBOL II Release 3 (5688-023).
- To use the double byte character set (DBCS) support, the correct compiler for COBOL or PL/I must be used:
  - VS COBOL II Release 2 (5688-023)
  - OS PL/I Version 2 Release 1 (5668-910).

CICS/ESA Version 4 supports IBM SAA AD/Cycle Language Environment/370 (5688-198) Version 1.1 and Version 1.2, and all compilers that require this library for run-time support. In addition, Language Environment/370 supports, in compatibility mode, programs compiled with VS COBOL II and C/370.

ISC and MRO communication

CICS/ESA 4.1 supports Intersystem communication (ISC) links with:

- Other CICS/ESA 4.1 regions
- CICS/ESA Version 3 regions
- CICS/MVS Version 2 regions
- CICS/OS/VS Version 1 Release 7 regions
- CICS/DOS/VS Version 1 Release 7 regions
- CICS OS/2
- CICS/400
- CICS/6000
- IMS/ESA Transaction Manager Version 3 Release 1
- Any system that supports advanced program-to program communication (APPC) protocols (LUTYPE6.2). For example, APPC/PC or AS/400.

CICS/ESA 4.1 supports multi region operation (MRO) for communication with:

- Other CICS/ESA 4.1 regions
- CICS/ESA Version 3 regions
- CICS/MVS Version 2 regions
- CICS/OS/VS Version 1 Release 7 regions

If the CICS regions are using the CICS/ESA 4.1 level of the DFHIRP module, and are running on MVS/ESA SP 5.0 or later, you can use XCF/MRO to communicate betweem MVS images.

Note: The function provided on any MRO or ISC connection is that of the lower release in the connection.

For CICS/ESA 4.1 to communicate with CICS at another release, the following APARs must be applied to the other CICS:

For CICS/MVS 2.1.0

- PL15145
- PL22728

The following APARs are also recommended for application for CICS/MVS 2.1:

- PL34825
- PL38004

Note: All the above APARs to be applied to CICS/MVS 2.1.0 were incorporated into CICS/MVS 2.1.1 and later CICS/MVS Version 2 releases.

For CICS/DOS Version 1.7

- PL15135

To enable CICS/ESA 4.1 regions to accept incoming transactions from releases of
CICS earlier than CICS/ESA 3.2.1, apply the CICS/ESA 4.1 PTF UN77613 for
APAR PN 63960. For related information, see the *CICS/ESA CICS-RACF Security*
Guide.

External CICS interface

Client programs running in an MVS address space can communicate only with CICS server regions running under CICS/ESA 4.1 or a later, upward-compatible release. This is because of the changes to the MRO connection definition to support the external CICS interface.

Also, the client program can connect to the server CICS region only through the CICS/ESA 4.1, or later, interregion communication program, DFHIRP.

Modules eligible for the MVS linklist

Modules that must be included in an APF-authorized library in the MVS linklist are installed in the SYS1.CICS410.SDFHLINK library. You ***must*** add these modules to an APF-authorized library in the MVS linklist, either by adding the SYS1.CICS410.SDFHLINK library to the MVS linklist or by adding the modules to another library in the MVS linklist. For further information about adding CICS modules to the MVS linklist, see the *CICS/ESA 4.1 Installation Guide*.

Modules eligible for the MVS link pack area (LPA)

Modules that must be included in the LPA are installed in the SYS1.CICS410.SDFHLPA library. Other CICS modules that can be included in the LPA are listed in the sample SMP/E usermods, installed in CICS410.SDFHSAMP. The member names for the sample usermods are:

DFH$UMOD
DFH$UJPN (for modules of the Japanese language feature)

For further information about adding CICS modules to the LPA, see the *CICS/ESA 4.1 Installation Guide*.

Application program compatibility

CICS/ESA 4.1 provides upward compatibility from CICS/ESA Version 3, CICS/MVS Version 2, and CICS/OS/VS Version 1.7 at both source and object level for CICS application programs and maps, subject to the exceptions and comments summarized below and further described in the *CICS/ESA Release Guide*, GC33-0655. Any other exceptions that are identified are documented in the CICS/ESA 4.1 publications.

- Command-level programs are upward compatible at both source and object level, provided they conform to the interface as defined in the *CICS/MVS 2.1.2*

Application Programmer's Reference manual, SC33-0512, and provided the function is still supported. However, support for the EXEC CICS ADDRESS CSA command has been discontinued.

- Customers who need to reassemble any command-level program written in Assembler language for CICS/ESA 4.1, that was originally assembled on CICS/MVS Version 2 or CICS/OS/VS Version 1, need to be aware that an additional base register may be required. This occurs only if the size of DFHEISTG was close to a 4096 addressing boundary on the earlier version of CICS, such that the extra storage that CICS/ESA Version 3 (or later) uses for parameter lists (136 bytes) causes a user variable to be moved past the boundary.
- Support is retained for CALL DL/I statements as well as EXEC DLI
- Macro-level program support has been discontinued. Applications that are not converted to command level should be run on CICS/MVS 2.1.2.

  The DFHMSCAN utility program, which is available with CICS/MVS Version 2 and CICS/ESA Version 3, or DFHMS170, which is available with CICS/OS/VS 1.7, is recommended for reviewing CICS application program libraries. This program can be run against each application load library to find out which application programs use CICS macros. The DFHMSCAN utility program provided with CICS/ESA Version 3 can also be used to check on use of the EXEC CICS ADDRESS CSA command.

  The CICS Application Migration Aid (5695-061), should be used to assist customers migrating macro-level programs to command-level programs.
- Basic mapping support (BMS) maps that are defined using CICS-supplied macro instructions, or defined online using Screen Definition Facility II (5664-366), are upward compatible.

Systems programming considerations

CICS/ESA Migration Guide, GC33-1162, includes guidance and details about operational and systems programming procedures introduced in CICS/ESA 4.1. CICS/ESA 4.1 provides upward compatibility, at source and object level for CICS application programs that are written to the system programming interface, and which executed correctly under CICS/ESA 3.3, except where otherwise stated in the *CICS/ESA Migration Guide*.

Also, if you are migrating to CICS/ESA 4.1 from a release of CICS earlier than CICS/ESA 3.3, you should obtain a copy of the *CICS/ESA Migration Guide for CICS/ESA Version 3 Releases 1 and 2*, GC33-0656.

The following functions previously available in CICS/ESA Version 3 are discontinued:

- Direct addressing of CICS control blocks, even for exits. (The application programming interface and system programming interface are enhanced to provide access to appropriate, required, CICS control information, and to enable manipulation of such information.)
- Control of maximum tasks by the system initialization parameters CMXT and CMXTLIM is replaced by resource definitions for transaction class objects.
- The signon table, DFHSNT.

- Support for the PROTECT option for VTAM terminals. After CICS/ESA 4.1, full protection of application and business logic will be supported by APPC functions.
- The BINDPASSWORD option on CONNECTION resource definitions.

The following functions will be not be available after CICS/ESA 4.1:

- Support for local DL/I. Database access will continue to be supported by the DBCTL interface.
- Support for logging or journal output to tape. Also, the formats of several of the system log record types will change.

The following functions previously available in CICS/MVS Version 2 and CICS/OS/VS 1.7 are discontinued:

- Support for macro-level execution
- Support for devices and controllers accessed using BTAM, GAM, or TCAM (ACB)
- Direct addressing of CICS control blocks (other than the EIB and user areas such as CWA) from within CICS applications
- EXEC CICS ADDRESS CSA command
- CICS internal security
- DFHXSP and DFHXSE as user-replaceable modules
- CSMT, CSSF, CSSN, CSST, and CSOT transactions
- System initialization modifications (SIMODs)
- PCTs, PPTs, and TCTs generated by resource definition macros. You must migrate macro-generated tables to the CSD, at the earlier release of CICS.

  TCT definition for TCAM (DCB), remote BTAM devices, and sequential terminals must be done with TCT macros.

Performance considerations

This section outlines performance considerations for CICS/ESA 4.1.

Throughput

1. ITR is equivalent to CICS/ESA 3.3 when no tracing is active. With standard tracing active there will be a 1% ITR improvement over CICS/ESA 3.3.
2. MRO/XCF can now be used for function shipping and transaction routing between CICS regions in a Sysplex. This will give a performance improvement over the alternative LU 6.2 method. This improvement is still to be measured.
3. Transaction isolation if turned on will result in a performance degradation in ITR of up to 5%.
4. When using CICS/ESA 4.1 with DB2 3.1 the new adaptor will give up to a 4% improvement in ITR.

Storage use

CICS/ESA 4.1 has four dynamic storage areas (DSAs) below the line and four dynamic storage areas in extended storage. Management of the DSAs has been simplified in CICS/ESA 4.1. Only one parameter (DSALIM) need be used to control the DSAs below the line, compared with four parameters in CICS/ESA 3.3. Only one parameter (EDSALIM) need be used to control the DSAs above the line compared to six parameters in CICS/ESA 3.3.

CICS usage of virtual storage below the 16Mb line has changed with the introduction of more self managed DSAs within an overall user specified limit. Initial storage comparisons indicate that the total virtual storage requirement below the 16Mb line for CICS/ESA 4.1 is equivalent to CICS/ESA 3.3.

However, when using the Transaction Isolation facility the User DSA is megabyte boundary aligned and an integral number of megabytes in length which may be less flexible for some workloads than when not using that facility.

When using transaction isolation there is a requirement of one megabyte EUDSA (Extended User Dynamic Storage Area) per concurrent active user task compared to 64K per task when not using transaction isolation. Initial storage comparisons indicate that the total virtual storage requirement for CICS/ESA 4.1 when using transaction isolation is similar to CICS/ESA 3.3 when using subsystem storage protection.

CICS components in object-code-only form

Some of the functional areas in CICS/ESA 4.1 are provided, either completely or partially, in object-code only form (OCO), without licensed source materials. These areas include:

- Autoinstall terminal model manager, AITM
- Catalog domains
- Common Programming Interface functions
- Data tables
- Dispatcher domain
- Directory manager domain
- EXEC CICS system programming command support
- Kernel domain
- Loader domain
- Lock manager domain
- Message domain
- Monitoring domain
- Program manager domain
- Offline statistics utility
- Offline system dump formatting routines
- Parameter manager domain
- Partner resource manager
- RDO for VSAM files and LSR pools
- SAA communications and resource recovery interfaces
- Security domain
- Statistics domain
- Storage manager domain
- Timer domain.
- Transaction manager domain

+ • User domain

Index

Numerics

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Sending your comments to IBM

CICS for MVS/ESA

Release Guide

GC33-1161-04

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 request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to 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, use the Readers' Comment Form
- By fax:
  - From outside the U.K., after your international access code use 44 962 870229 (after 16 April 1995, use 44 1962 870229)
  - From within the U.K., use 0962 870229 (after 16 April 1995, use 01962 870229)
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: WINVMJ(IDRCF)
  - Internet: idrcf@winvmj.vnet.ibm.com

Whichever you use, ensure that you include:

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

Readers' Comments

CICS for MVS/ESA

Release Guide

GC33-1161-04

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

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.

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Telephone	Email

You can send your comments POST FREE on this form from any one of these countries:

Australia	Finland	Iceland	Netherlands	Singapore	United States
Belgium	France	Israel	New Zealand	Spain	of America
Bermuda	Germany	Italy	Norway	Sweden	
Cyprus	Greece	Luxembourg	Portugal	Switzerland	
Denmark	Hong Kong	Monaco	Republic of Ireland	United Arab Emirates	

If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

1 Cut along this line

2 Fold along this line

By air mail
Par avion

IBRS/CCRI NUMBER: PHQ - D/1348/SO

NE PAS AFFRANCHIR

NO STAMP REQUIRED

REPONSE PAYEE
GRANDE-BRETAGNE

IBM United Kingdom Laboratories
Information Development Department (MP095)
Hursley Park,
WINCHESTER, Hants
SO21 2ZZ United Kingdom

3 Fold along this line

From: Name _____

Company or Organization _____

Address _____

_____

EMAIL _____

Telephone _____

1 Cut along this line

4 Fasten here with adhesive tape

Program Number: 5655-018

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

GC33-1161-04